OFFICE OF THE SECRETARY OF DEFENSE
OPERATIONAL TEST AND EVALUATION
1700 DEFENSE PENTAGON
WASHINGTON, DC 20301-1700

DEPARTMENT OF THE ARMY
US ARMY MATERIEL COMMAND
ABERDEEN PROVING GROUND, MD 21005-5071

DEPARTMENT OF THE NAVY
ASSESSMENT DIVISION OPNAV (N81)
WASHINGTON, DC 20350-2000

DEPARTMENT OF THE AIR FORCE
HEADQUARTERS AIR ARMAMENT CENTER (AFMC)
EGLIN AFB, FL 32542-6808

JOINT TECHNICAL COORDINATING GROUP FOR MUNITIONS EFFECTIVENESS
ABERDEEN PROVING GROUND, MARYLAND 21005-5071

RDAM-J 5 Aug 10

MEMORANDUM FOR Joint Technical Coordinating Group for Munitions Effectiness Product Management Office (OC-ALC/ENLB/Ms. Sandra Hysell), 7851 Arnold Street, Suite 202, Tinker AFB, OK 73145-9160

SUBJECT: Distribution Statement for 61 JTCG/ME-71-7-1, 61 JTCG/ME-7-2-1 and 61 JTCG/ME-71-7-2-2

1. A review of the subject Magic Computer Simulation User and Analyst Manuals has been conducted based upon a request received from the US Army Research Laboratory. This review resulted in the decision to release these publications into the public domain. As such, request the following distribution statement be added to these items: "Approved for public release; distribution is unlimited."

2. Request, therefore, recipients of these publications be notified of this distribution statement.

3. The JTCG/ME Program Office point of contact for this request is Mrs. Chantal B. Marus, COMM (410) 278-6740, DSN 298-6740; e-mail: chantal.b.marus@us.army.mil.

[illegible]

BRYAN W. PARIS
Director
JTCG/ME Program Office

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **MAY 1971**	2. REPORT TYPE **Final**	3. DATES COVERED **00-00-1969 to 00-00-1971**
4. TITLE AND SUBTITLE **MAGIC Computer Simulation Analyst Manual Part 1**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Weapons Center,China Lake,CA,93555**		8. PERFORMING ORGANIZATION REPORT NUMBER **TN4565-3-71 Vol II**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **Joint Technical Coordinating Group for Munitions Effectiveness, ABERDEEN PROVING GROUND, MD, 21005-5071**		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S) **61 JTCG/ME-71-7-2-1**
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		

14. ABSTRACT

The MAGIC computer simulation generates target description data consisting of item-by-item listings of the target’s components and air-spaces encountered by a large number of parallel rays emanating from any desired attack angle. A combinatorial geometry technique, which defines the locations and shapes of the various physical regions in terms of the intersections and unions of the volumes contained in a set of simple bodies, is used to represent complex target structures. A grid cell pattern is superimposed over the surface of the target and parallel rays are "fired" from each grid cell.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**452**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

61 JTCG/ME-71-7-2-1

JTCG/ME

MAGIC COMPUTER SIMULATION

VOLUME II. ANALYST MANUAL

PART I

Produced for:

Joint Technical
Coordinating Group
for
Munitions Effectiveness

MAY 1971

ABSTRACT

The MAGIC computer simulation generates target description data consisting of item-by-item listings of the target's components and air-spaces encountered by a large number of parallel rays emanating from any desired attack angle. A combinatorial geometry technique, which defines the locations and shapes of the various physical regions in terms of the intersections and unions of the volumes contained in a set of simple bodies, is used to represent complex target structures. A grid cell pattern is superimposed over the surface of the target and parallel rays are "fired" from each grid cell.

Volume II, Part I contains:

Section I	Introduction
Section II	Mathematical Model
Section III	Simulation Model, MAIN Program through Subroutine TOR.

ACKNOWLEDGEMENT

To assist in the evaluation of the vulnerability of armored vehicles, Mathematical Applications Group, Inc. of White Plains, New York developed the MAGIC Computer Simulation under Contract DAAD05-67-C-0041 for the Department of the Army, Ballistic Research Laboratories, Richard C. Hoyt, Technical Supervisor. Previously published documents--A Geometric Description Technique Suitable for Computer Analysis of Both Nuclear and Conventional Vulnerability of Armored Military Vehicles, AD 847576, August 1967, and The MAGIC-SAM C Target Analysis Technique, Volume VI, AMSAA TR 14, April 1969--are updated by the User and Analyst Manuals written by Armament Systems, Inc., Anaheim, California.

These new manuals reflect the current state of the art and provide for future documentation maintenance on a page-by-page basis.

Mr. L. W. Bain, Army Material Systems Analysis Agency, and Mr. M. J. Reisinger, Ballistic Research Laboratories provided technical coordination during the preparation of the new documents.

The work was performed under Naval Weapons Center, China Lake, California, Contract N00123-70-C-0576, administered by Mr. H.W. Drake, Propulsion Development Department. The documentation conforms to the format specified by NWC Technical Note 4565-3-69, Volumes I and II.

The Armament Systems, Inc. personnel responsible for preparation were John E. Musch and Robert A. Burris.

Publication is carried out under the auspices of the Air Target Vulnerability Sub-Group of the Joint Technical Coordinating Group for Munitions Effectiveness (JTCG/ME). The Sub-Group membership includes:

Frank W. Sieve - Chairman, Naval Ordnance Systems Command

Donald W. Mowrer - Army Member, U. S. Army Ballistic Research Lab.

Orlando T. Johnson - Army Member, U. S. Army Ballistic Research Lab.

1/Lt. Abraham A. Santiago, Jr. - Air Force Member, Eglin AFB

Gerald B. Bennett, Jr. - Air Force Member, Wright Patterson AFB

Hubert W. Drake - Navy Member, Naval Weapons Center

W. Wallace Morton, Jr. - Navy Member, Naval Weapons Laboratory

Robert E. Gray - Chairman, Methodology Working Group, Naval Weapons Laboratory

SUMMARY

The MAGIC computer simulation generates target description data with the detail and completeness required for vulnerability studies. A combinatorial geometry technique is used in the simulation to represent a complex target structure. A large number of parallel rays, randomly located in grid cells, are traced through the target structure to produce item-by-item listings of the components and air spaces.

COMBINATORIAL GEOMETRY TECHNIQUE

The basic technique for a geometry description consists of defining the locations and shapes of the target physical regions (wall, equipment, etc.) utilizing the intersections and unions of the volumes of twelve simple body shapes. A special operator notation uses the symbols (+), (-), and (OR) to describe the intersections and unions. These symbols are used by the program to construct tables used in the ray-tracing portion of the program.

GEOMETRICAL DESCRIPTION

The user specifies the type and location of each body used to describe the target; and identifies physical regions in terms of these bodies. Each region is assigned an identification code for use with vulnerability analyses. A three-dimensional coordinate system is established in relation to the target, which is enclosed by a rectangular parallelepiped. A grid plane is established according to the attack angle desired, and parallel rays, starting randomly from each grid cell, are traced through the target.

INPUT

In the normal operating mode, target description data is input by cards. A portion of the routine converts the data to the form required for ray-tracing. The input data is checked; if errors are detected, messages are printed out. Error-free target description data may then be stored on magnetic tape and input in this form on subsequent production mode operations.

OUTPUT

The basic output is the results of the ray-tracing computations. A listing is obtained, for each grid cell, of the line-of-sight thickness for each geometrical region traversed, the obliquity of the ray with respect to the normal of the first surface of each region encountered, and the normal distance through each region.

OPTIONAL ROUTINES

Three optional routines are available to the user: special ray tracing used for target data checking; region volume calculations; and computing target presented area.

PROGRAMMING

The simulation, which is programmed using FORTRAN, requires a large-scale digital computer. The simulation is currently operational on both the CDC-6600 and BRL-BRLESC computers.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Concluded)

LIST OF TABLES

SECTION I

INTRODUCTION

The MAGIC computer simulation generates target description data with the detail and completeness required for vulnerability studies. The target description data consists of item-by-item listings of the components and air spaces encountered by a large number of parallel rays emanating from any attack angle and passing through any type of target.

A combinatorial geometry technique is used to represent a complex three-dimensional target structure, such as a tank, in terms of sums, differences, and intersections of relatively simple bodies. The input for such a description consists of the geometric location and dimensions of the simple bodies followed by a region definition table consisting of a series of equations defining each region in terms of the simple bodies. In addition to the geometric description, a coded number is assigned to each region to identify its function.

The computer routine superimposes a grid cell pattern over the surface of the target, as viewed from the attack angle desired, randomly locating parallel rays in each grid cell. The computer traces each ray through the target; and each target item encountered is listed sequentially and identified as to ray location in the grid, target identification, line-of-sight thickness, normal thickness, angle of obliquity, identification of the air space following the target, and line-of-sight distance through the air space.

COMBINATORIAL GEOMETRY TECHNIQUE

The combinatorial geometry technique has been developed to produce a model that is both accurate and suitable for a ray-tracing analysis program. The basic technique for a geometry description requires defining the locations and shapes of the various physical regions (wall, equipment, etc.), utilizing the intersections and unions of the volumes of twelve simple bodies. The geometric bodies are as follows:

(1) Rectangular parallelepipied

(2) Box

(3) Sphere

(4) Right circular cylinder

(5) Right elliptical cylinder

(6) Truncated right angle cone

(7) Ellipsoid

(8) Right angle wedge

(9) Arbitrary convex polyhedron of four, five, or six sides

(10) Truncated elliptic cone

(11) Arbitrary surface

(12) Torus

A special operator notation uses the symbols (+), (-), and (OR) to describe the intersections and unions. These symbols are used by the program to construct tables used in the ray-tracing portion of the problem. If a body appears in a region description with a (+) operator, the region being described is wholly contained in the body. If a body appears in a region description with a (-) operator, the region being described is wholly outside the body. A region may be described in terms of several subregions lumped together by (OR) statements.

The technique of describing a physical region is best illustrated by examples. Imagine a mallet consisting of two cylinders. Call the mallet head solid number 1 and the handle solid number 2.

The two cylinders may be physically positioned and logically described in several ways. One way is to consider the handle and head as separate regions, as shown in Figure 1. The region description is region 1 = 1 and region 2 = 2-1.

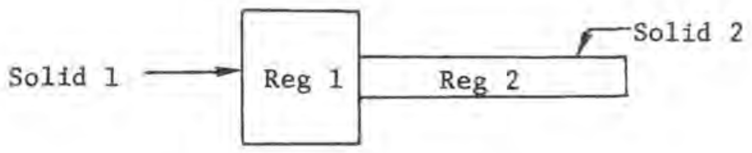


FIG. 1. Mallet with Handle and Head as Separate Regions

Another way is to think of the handle extending into the head, as shown in Figure 2. A logical method of describing this mallet is region 1 = 1-2 and region 2 = 2, indicating that the mallet head contains a cylindrical hole into which the handle is fitted.

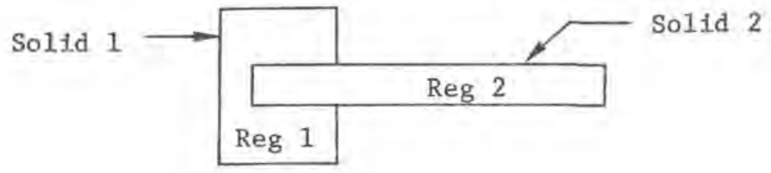


FIG. 2. Mallet with Handle Extending Into the Head

Now consider a description of a mallet physically similar to that in Figure 2 but whose handle consists of two types of material, one outside the mallet head and the other inside the head, as shown in Figure 3. A logical way to describe this is region 1 = 1-2, region 2 = 2-1, and region 3 = 1+2.

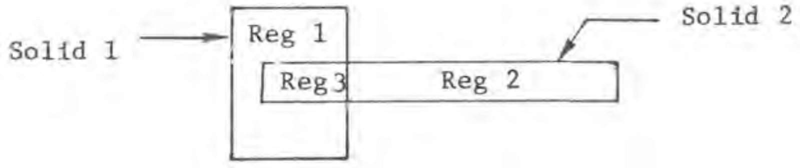


FIG. 3. Mallet with Handle Consisting of Two Types of Materials

A fourth way is to lump the mallet head and handle into one region, considering them to be like materials, as shown in Figure 4. The description then is region 1 = (OR) 1 (OR) 2. This means that a point in region 1 may be in either solid 1 or solid 2.

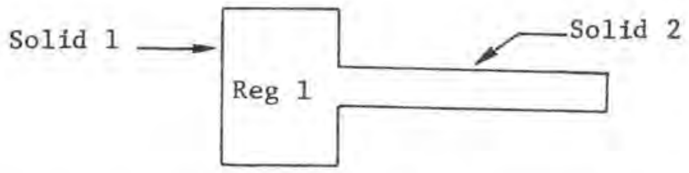


FIG. 4. Mallet with Head and Handle of Like Materials

A rule of construction imposes the additional restriction that region descriptions include negation (-) of buttressing surfaces not otherwise necessary to the logical description of the region. That is, if points on the surface of body 2 are common to parts of the surface of body 3, as shown in Figure 5, the description of region 200 is 200 = (+2) (-3). Region 300 is defined as 300 = (+3) (-2).

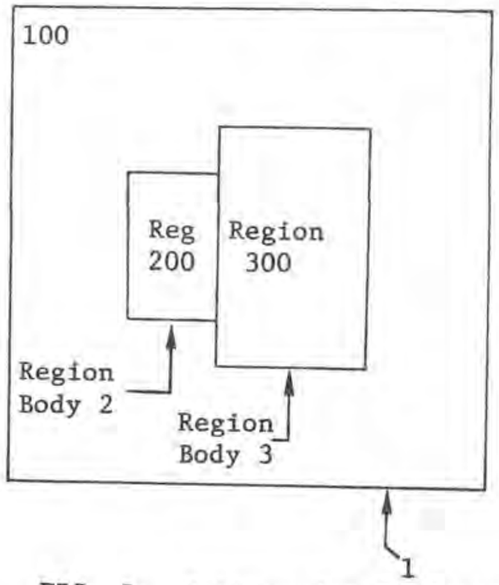


FIG. 5. Buttressing Surfaces

GEOMETRICAL DESCRIPTION

The user of the program must specify the geometrical description by establishing two tables. The first table describes the type and location of the set of bodies to be used. The second table identifies the physical region in terms of these bodies. The computer program converts these tables into the form required for ray tracing. Note well: all the space must be divided into regions, and no point may be in more than one region.

Coordinate System

The geometric figures used to define the target are located relative to one another by the use of a three-dimensional coordinate system superimposed on available engineering drawings. A readily identifiable reference point should be designated from which the three-dimensional coordinate system can easily be constructed. On armored vehicles such as tanks, the intersection of the turret datum line and the center lines of the turret forms a natural reference point for the coordinate system origin as illustrated by the simplified tank in Figure 6.

Rectangular Parallelepipeds (RPP)

Once the coordinate system is established, the target is inclosed in an environment consisting of rectangular parallelepipeds (RPP's). The RPP's are solid geometric figures used for gross subdivisions of the target environment, which consists of the air surrounding and the ground under the target.

Twelve RPP's are used for the nuclear analysis of targets, as shown in Figure 7, but only one RPP is required for conventional target analyses. Twelve RPP's should be considered for all target descriptions so as to standardize the target descriptions for use with either conventional or nuclear analyses.

Identification Codes

Each region is assigned an identification code for use with conventional vulnerability and MAGIC programs. A three-digit code is assigned to each component of the target, such as armor, gun tube; and a two-digit code is assigned to each space, such as inside air, outside air. A general division of identification codes might be as shown in Table 1. A component described using more than one region will have its ID assigned to each region.

Grid

All the rays that are traced through the target geometry originate in the grid plane, which is a plane divided into equal squares called grid cells and oriented so that a ray passed perpendicularly from the center of the plane

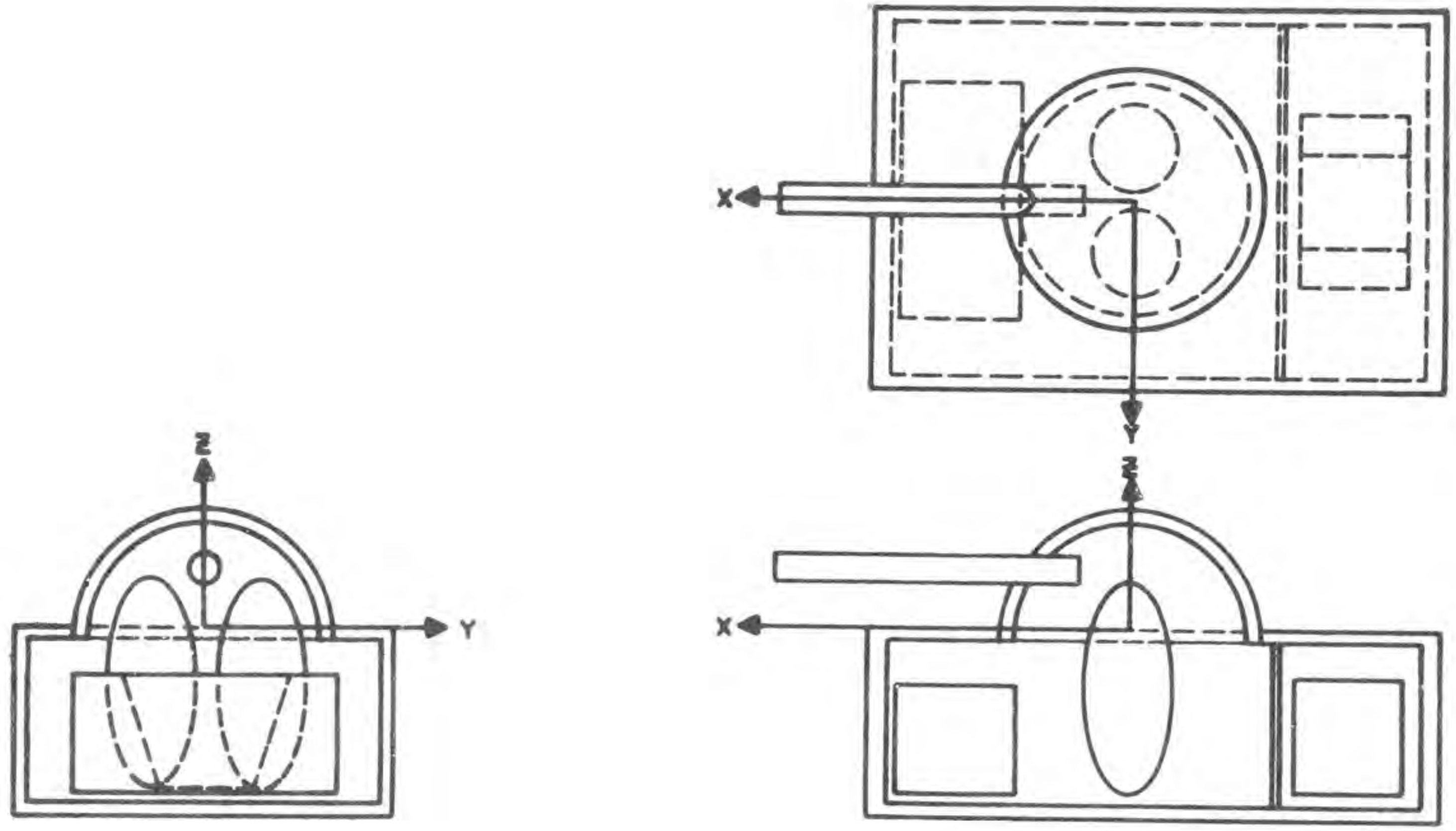


FIG. 6. Coordinate System Superimposed on Simplified Tank

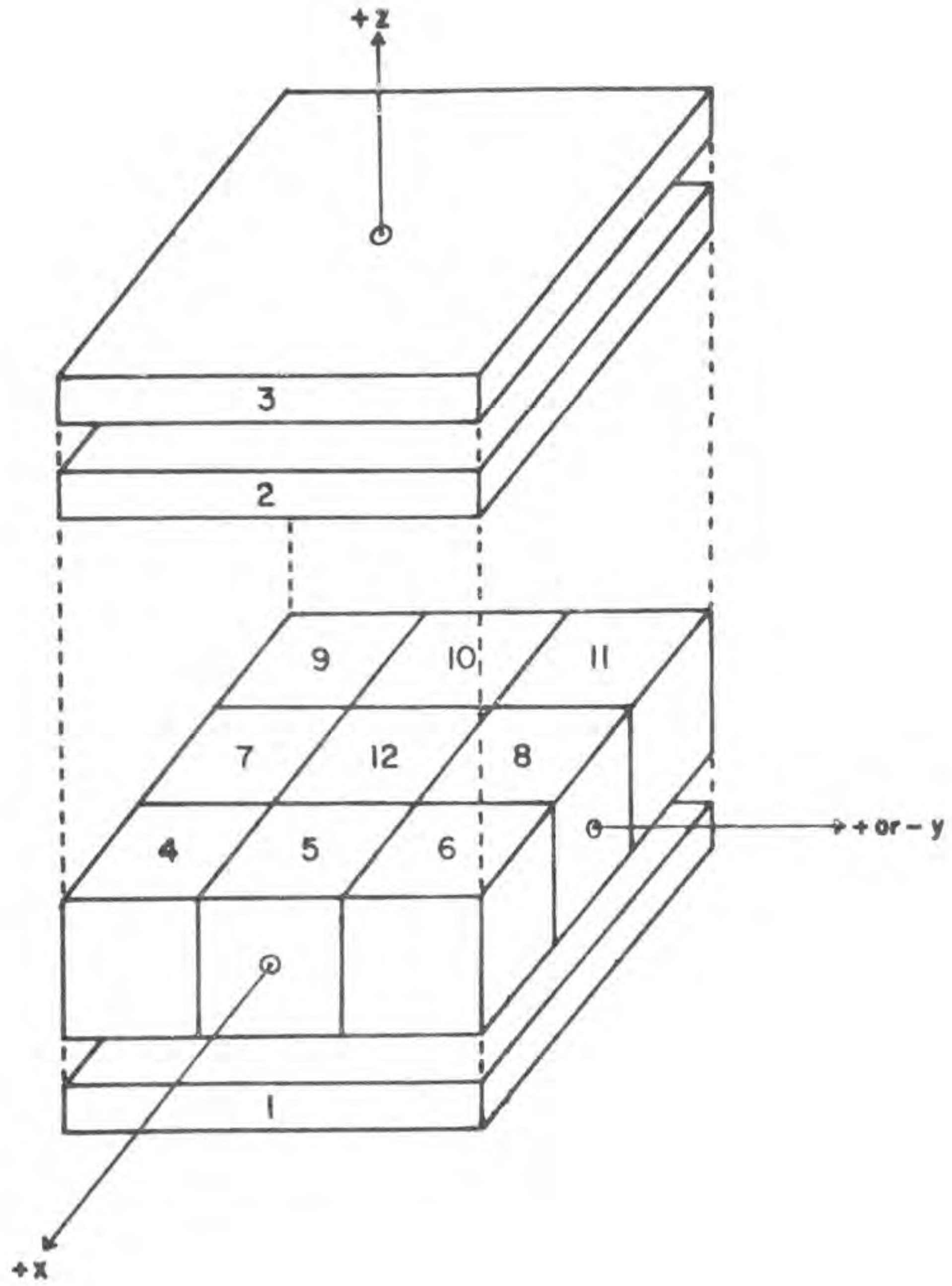


FIG. 7. Twelve RPP's

TABLE 1. Identification Codes

Component Codes

ID Codes	Type of Component
001-099	Internal components
100-199	Types of armor
200-299	Fuel components
300-399	Miscellaneous exterior components
400-499	Gun components
500-599	Track suspension components*
600-699	Wheel suspension components
700-799	Power train components
800-899	Miscellaneous components
900-998	Not used at present
999	The ground

*ID Code 501 is reserved for the track; the computer assigns 502 if the track edge is hit.

Space Codes

Space Number	Type of Space
00	Not used at present
01	External air
02	Crew compartment air
03	Not used at present
04	Not used at present
05	Engine compartment air
06	Not used at present
07	Not used at present
08	Not used at present
09	No further target

NOTE: The operation of the MAGIC program will not allow assigning different space codes to bounding regions. In other words, a ray passing through the geometry cannot pass directly from outside air (01) to inside air (02). There must be a three-digit coded item between different space regions.

to the target will intersect the target coordinate system origin. The grid plane is established with the following information: grid size, attack angle of the target, and back-off distance from the origin of the coordinate system.

The attack angle is specified in terms of an azimuth and elevation angle using a right-handed coordinate system. A positive azimuth angle is measured from the positive X axis in a counterclockwise direction when viewed from above, as shown in Figure 8. Elevation angles are measured from the X-Y plane positive upward.

The back-off distance is the distance from the origin of the coordinate system used in the target description to the grid plane. All the rays originating from a common grid plane must start in the same region; therefore, the grid plane must be placed within the bounds of one region. If the grid plane is to include the entire target, it must lie outside the target as described in the region description. If only a certain component of the target is to be considered (for instance, the engine of a tank), care must be taken to insure that the grid plane lies outside the engine as described, that it lies within the bounds of only one region, and that all rays end in a common region.

Cellular Output

The basic output of the MAGIC simulation consists of cellular output obtained from the ray tracing computations. The ray tracing phase is the process whereby rays (one for each cell) are traced perpendicularly from the grid plane through the target geometry. The calculated output for each ray consists of the line-of-sight thickness of each geometric region traversed, the obliquity (angle of incidence) of the ray with respect to the first surface of each region encountered (excluding air or spaces), and the normal or perpendicular distance through each region from the point of entry (excluding air or spaces). One unique feature of the program is that thicknesses of bounding geometric regions with like functional identifiers are cumulative. A representative vehicle section for target cell description data is shown in Figure 9.

Data Input Error

The simulation contains statements to check the validity of the target geometry data. Some of the errors are considered fatal and will cause execution to terminate, while others will be noted and special error messages printed. A tally is maintained of the non-fatal errors; if they exceed a specified number, execution terminates. Table 2 lists the error items and the subroutine in the simulation where the error check statement is located.

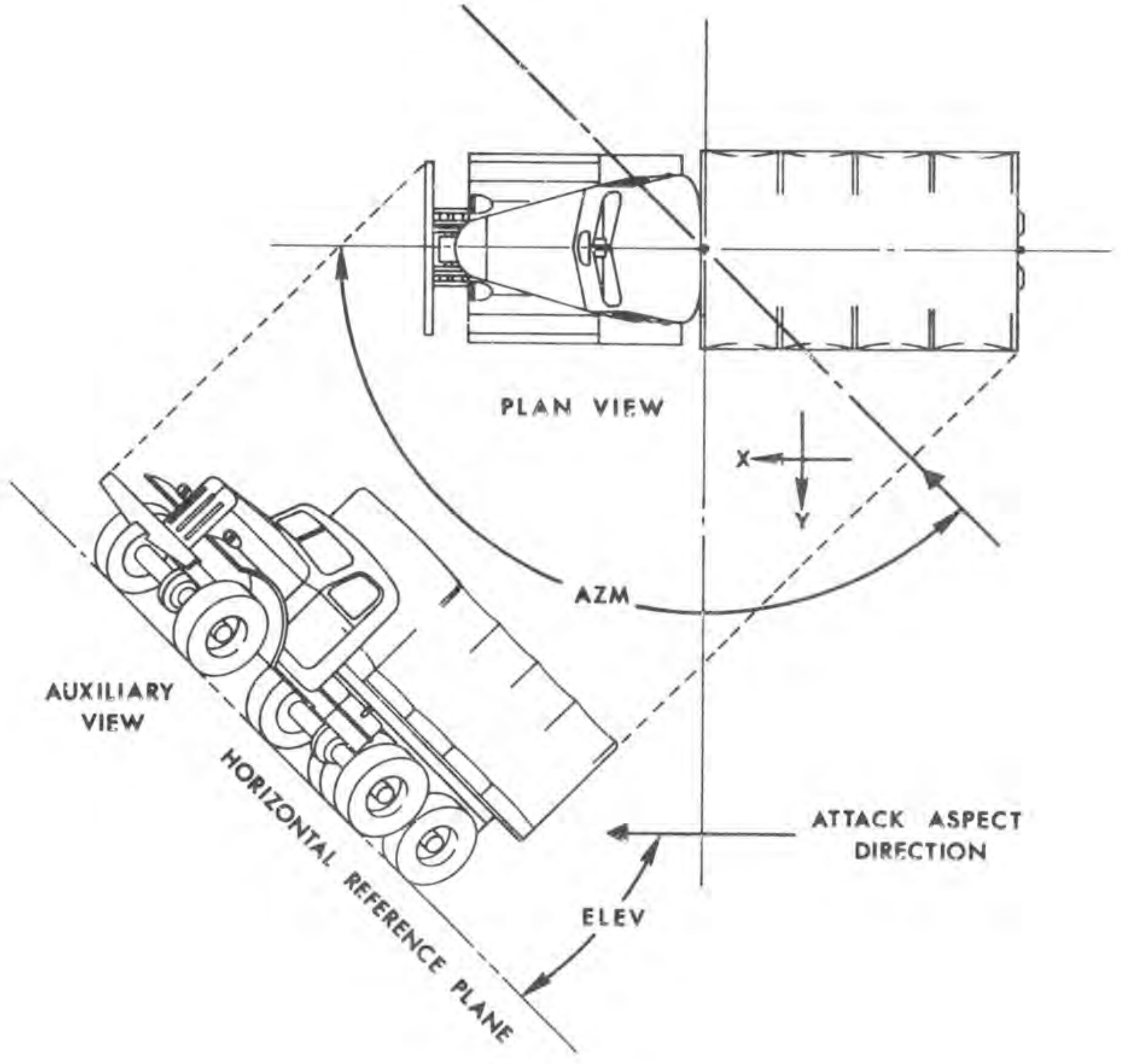


FIG. 8. Attack Angle Geometry

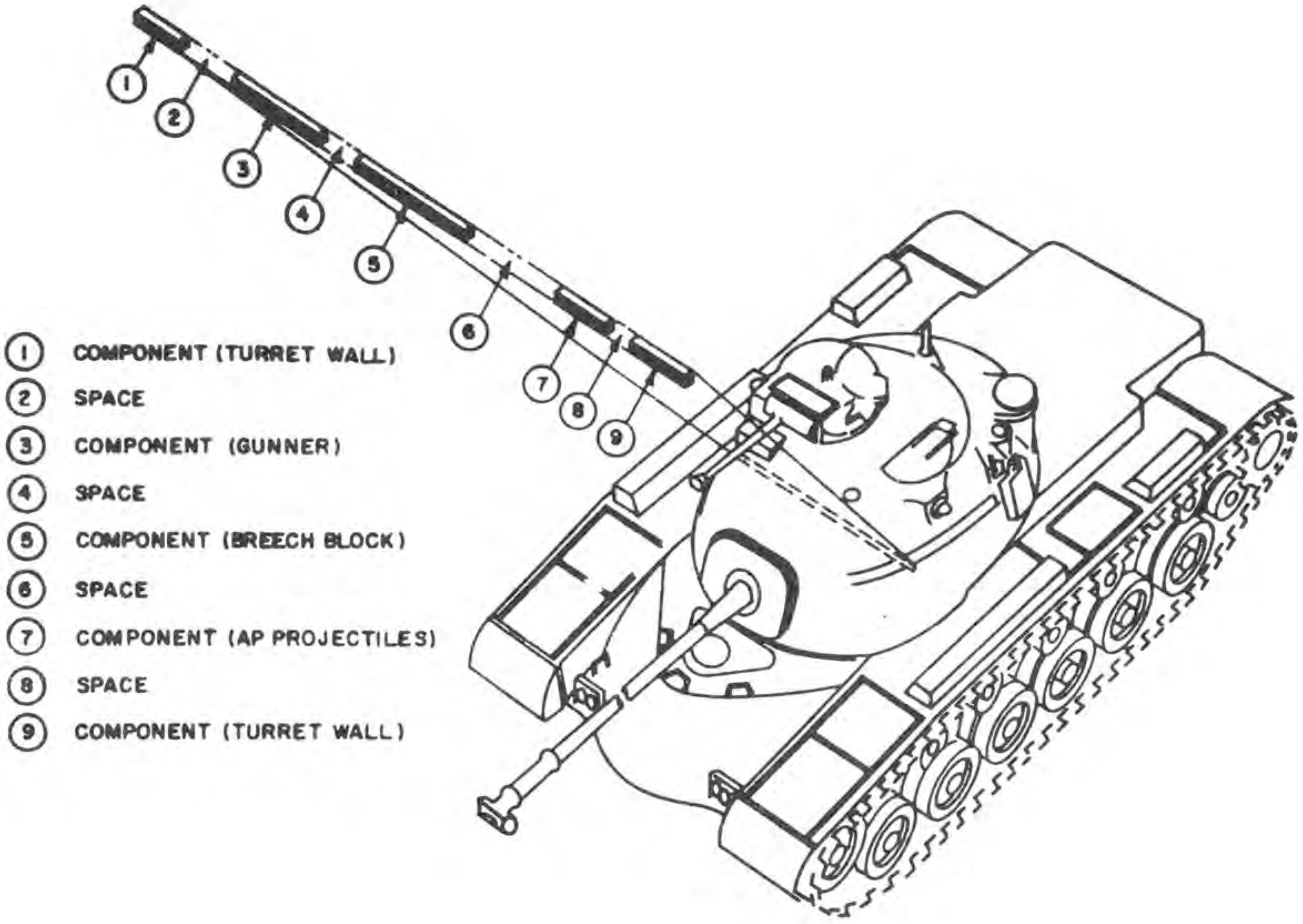

1 COMPONENT (TURRET WALL)
2 SPACE
3 COMPONENT (GUNNER)
4 SPACE
5 COMPONENT (BREECH BLOCK)
6 SPACE
7 COMPONENT (AP PROJECTILES)
8 SPACE
9 COMPONENT (TURRET WALL)

FIG. 9. Representative Vehicle Section for Target Cell Description Data

TABLE 2. Data Input Errors

Subroutine	Description	Error Type
GENI	Body card does not contain correct body abbreviation	Fatal
	Minor radius of torus equal to or greater than major radius	Non-fatal
	Semi-major and semi-minor axes of truncated elliptical cone are not perpendicular or height vector is parallel to base	Non-fatal
	Radii of upper and lower bases same for truncated elliptical cone	Non-fatal
	Vectors used to describe a box, right angle wedge, or truncated elliptical cone are not mutually perpendicular	Non-fatal
	Storage locations for body data exceed allowable value	Fatal
	Logical operator was not located	Fatal
	Storage locations for region data exceed allowable value	Fatal
	Number of regions in region table input does not match the number of regions specified	Fatal
	Number of body description cards does not match the number specified	Fatal
	Region description error	Fatal
	Storage for enter/leave table exceeded	Fatal
RPPIN	RPP description errors	Fatal

TABLE 2. (Concluded)

Subroutine	Description	Error Type
ALBERT	Undefined plane in arbitrary convex polyhedron (ARB) input	Non-fatal
	Four points describing a face of of ARB are not in a plane	Non-fatal
	Error in numering points of ARB	Non-fatal
VOLUM	Next region number negative	Fatal
MAIN	No storage available for region identification data	Fatal
CALC	Error in body type number	Fatal
	No normal found for arbitrary surface	Non-fatal
G1	Error in body type number	Non-fatal
	No intersect found in region	Non-fatal
	Error in body number of intersected RPP	Non-fatal
	Error in surface number of intersected RPP	Non-fatal
	No entries in region enter table	Non-fatal
	No region found for present point	Non-fatal
	Distance to next region is less than zero	Non-fatal
WOWI	Error in body type number	Non-fatal
ARS	Data in hit table is in error	Non-fatal
RPP	More than two surfaces of RPP were intersected	Non-fatal
AREA	Storage for area data exceeded	Fatal

OPTIONAL ROUTINES

Three optional subroutines--TESTG, VOLUM, and AREA--are available to the user for performing special computations.

Subroutine TESTG

This routine may be used to trace a specified number of rays in any portion of the target. These special computations are useful in checking the input data target geometry and region specifications.

Subroutine VOLUM

This routine may be used to compute the volume of each region contained within a specified portion of the target. An imaginary box is specified, and the volume of each region in the box is computed.

Subroutine AREA

This routine may be used to compute the presented area of the target as viewed from the specified attack angle. The presented area data is categorized according to the component identification number of the first component struck by the rays and to the total target.

SECTION II

MATHEMATICAL MODEL

This section presents the mathematical model employed in the simulation to perform the ray tracing calculations. Descriptions are provided for six major elements:

1. Intersection distance calculations involving the various geometrical shapes.
2. Normal distance calculations involving the various geometrical shapes.
3. Grid plane and ray starting point geometry.
4. Solutions to quartic and cubic equations.
5. Data arrangement in the principal (MASTER-ASTER) storage array.
6. Conceptual type flowcharts for the major control and input processing subroutines.

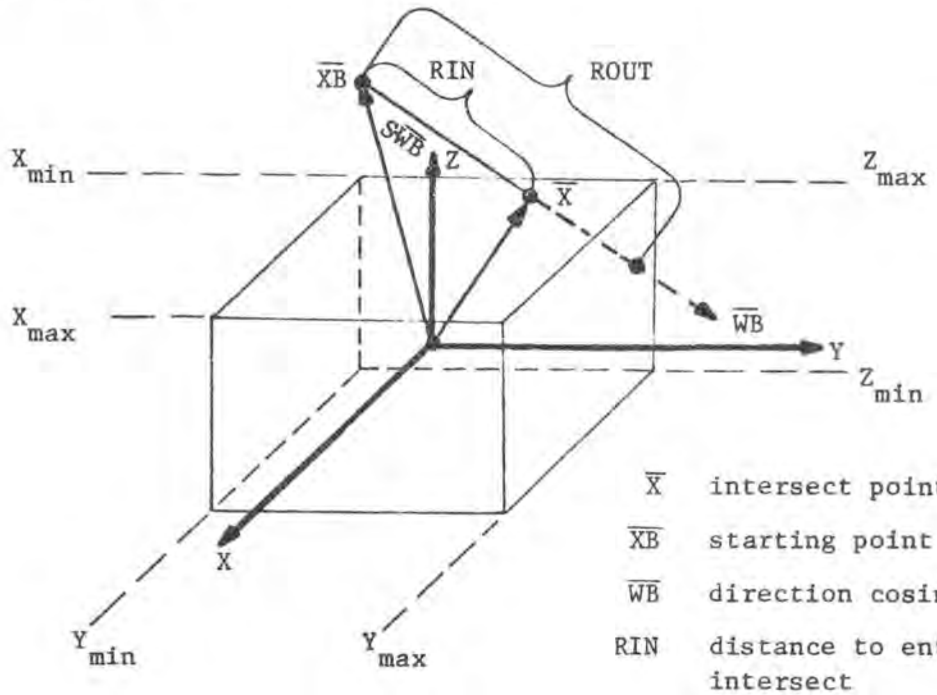
In each section the appropriate geometrical relationships are described, necessary assumptions and constraints delineated, and required algebraic relationships established.

The equations in the mathematical model that are boxed are the major equations that are used in the simulation model that gave rise to a FORTRAN statement. The same equation can be found in the appropriate section of the simulation model and is also shown as a boxed equation for ease of cross-referencing.

RECTANGULAR PARALLELEPIPED (SUBROUTINE RPP)

Subroutine RPP calculates the intersection distances (if any) of a ray in space and a rectangular parallelepiped. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equations defining the six bounding planes of the RPP.

The rectangular parallelepiped must have bounding surfaces parallel to the coordinate axes. The bounding planes of the RPP are therefore defined by specifying the maximum and minimum values of the x, y, and z coordinates as shown in Figure 10.



$\overline{X}$	intersect point on body
$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 10. Rectangular Parallelepiped

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB} \cdot S \qquad (1)$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (1) will give a point $\overline{X}$ (x,y,z) along the ray.

Therefore, the distance to any one of the six surfaces of the RPP can be expressed as

$$\boxed{S_I = \frac{\overline{XS}_I - \overline{XB}_J}{\overline{WB}_J}} \qquad (2)$$

J	I
1	1,2
2	3,4
3	5,6

where J = 1, 2, 3 for the x, y, and z coordinates, respectively, and I = 1-6 for the six surfaces of the RPP. The surfaces are denoted as follows.

$\overline{XS}_1 = X_{min}$ plane coordinate

$\overline{XS}_2 = X_{max}$ plane coordinate

$\overline{XS}_3 = Y_{min}$ plane coordinate

$\overline{XS}_4 = Y_{max}$ plane coordinate

$\overline{XS}_5 = Z_{min}$ plane coordinate

$\overline{XS}_6 = Z_{max}$ plane coordinate

When $\overline{WB}_J = 0$, the ray is parallel to the J pair of planes and no intersection will occur. When $S_I < 0$, the ray is moving away from the plane and no intersection with the plane will occur.

Knowing S_I, the distance from $\overline{XB}$ to the intersection of the plane, the y-z coordinates (assume an X plane for illustration) can be determined by ray Equation(1) as follows.

$$\boxed{\overline{XI}_C = \overline{XB}_C + \overline{WB}_C \cdot S_I} \qquad (3)$$

where C represents the y or z coordinate.

The given intersection, $\overline{XI}$, with an X plane is within the boundary of the RPP if the y-z coordinates of the intersect are within the Y and Z bounding planes. Therefore, $\overline{XI}$ (the intersection with an X plane) is within the boundary of the RPP when

$$\boxed{\begin{array}{l} Y_{min} \leq Y_C \leq Y_{max} \\ \\ Z_{min} \leq Z_C \leq Z_{max} \end{array}} \qquad (4)$$

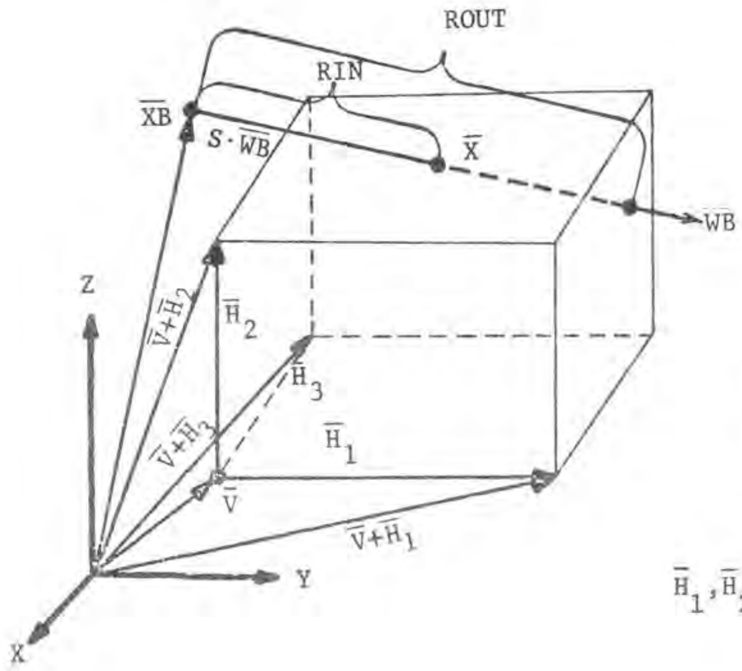
where Y_C and Z_C are the intersect coordinates of the X plane.

RIN is the minimum S_I that satisfies Equation (4), and ROUT is the maximum S_I that satisfies Equation (4). If there are no S_I intersects that satisfy Equation (4), ROUT is set to -10^{50} and RIN is set to 10^{50} to represent a miss of the RPP.

BOX (SUBROUTINE BOX)

Subroutine BOX calculates the intersection distances (if any) of a ray in space and a box. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equations defining the sides of the box.

The box is defined by a vertex, $\overline{V}$, and three mutually perpendicular length vectors. These three vectors represent the height, width, and length of the box as shown in Figure 11.



$\overline{X}$	intersect point of ray
$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	vertex of box
$\overline{H}_1, \overline{H}_2, \overline{H}_3$	three mutually perpendicular length vectors of the box
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 11. Box

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB} \cdot S \tag{5}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (5) will give a point $\overline{X}$ (x,y,z) along the ray.

For any pair of sides of the box, one side contains the vertex, $\overline{V}$, and the other side contains the point $\overline{V} + \overline{H}_i$, where i = 1, 2, or 3 for the length vector perpendicular to the pair of sides considered. The equation of the side containing $\overline{V}$ is given by the dot product

$$(\overline{V}-\overline{X}) \cdot \overline{H}_i = 0 \tag{6}$$

where $\overline{X}$ is any point on the side and i is equal to 1, 2, or 3 for the length vector perpendicular to the intersected sides. The equation of the opposite side of the box containing $\overline{V} + \overline{H}_i$ is given by the dot product

$$(\overline{V} + \overline{H}_i - \overline{X}) \cdot \overline{H}_i = 0 \tag{7}$$

The intersection distances of the ray with the box are obtained by substituting the $\overline{X}$ of Equation (5) into Equations (6) and (7) giving

$$\boxed{S^i_V = \frac{(\overline{V}-\overline{XB}) \cdot \overline{H}_i}{\overline{WB} \cdot \overline{H}_i}} \tag{8}$$

for the side containing $\overline{V}$ and

$$S^i_{V+H} = \frac{(\overline{V}-\overline{XB}) \cdot \overline{H}_i}{\overline{WB} \cdot \overline{H}_i} + \frac{\overline{H}_i \cdot \overline{H}_i}{\overline{WB} \cdot \overline{H}_i}$$

or

$$\boxed{S^i_{V+H} = S^i_V + \frac{\overline{H}_i \cdot \overline{H}_i}{\overline{WB} \cdot \overline{H}_i}} \tag{9}$$

for the side containing $\overline{V}+\overline{H}_i$. For a given pair of intersects, RIN is the smaller and ROUT the larger. From Equation (9)

$$S^i_{V+H} - S^i_V = \frac{\overline{H}_i \cdot \overline{H}_i}{\overline{WB} \cdot \overline{H}_i}$$

The sign of $S^i_{V+H} - S^i_V$ is determined by the sign of $\overline{WB} \cdot \overline{H}_i$ since $\overline{H}_i \cdot \overline{H}_i$ is always positive.

Therefore,

$$RIN_i = S^i_V \text{ and } ROUT_i = S^i_{V+H}$$

for

$$\overline{WB} \cdot \overline{H}_i > 0$$

or

$$RIN_i = S^i_{V+H} \text{ and } ROUT_i = S^i_V$$

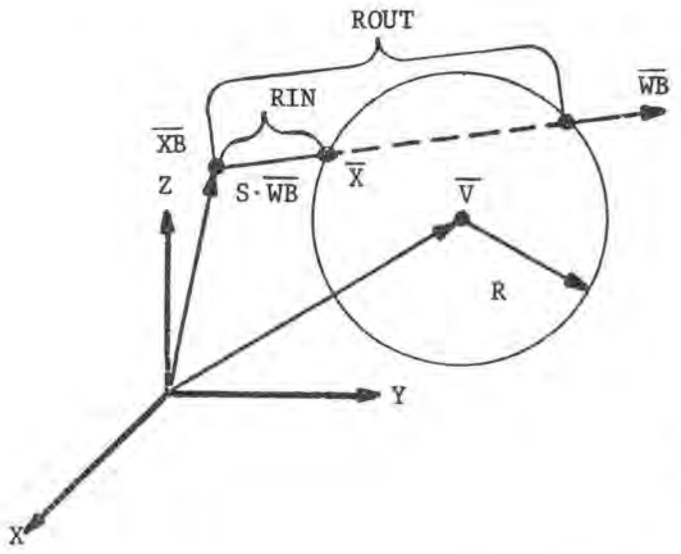
for

$$\overline{WB} \cdot \overline{H}_i < 0.$$

RIN is the maximum of the RIN_i, and ROUT is the minimum of the $ROUT_i$. For any $ROUT_i < 0$ the ray cannot intersect the box so ROUT is set to -10^{50} and RIN is set to 10^{50}.

SPHERE (SUBROUTINE SPH)

Subroutine SPH calculates the intersection distances (if any) of a ray in space and a sphere. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equation defining the sphere (see Figure 12).



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	center of sphere
R	radius of sphere
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 12. Sphere

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB}\cdot S \tag{10}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation(10) will give a point $\overline{X}$ (x,y,z) along the ray.

The equation of a sphere of radius R about the point $(\overline{V}_x, \overline{V}_y, \overline{V}_z)$ for any point $\overline{X}$ on the surface is

$$(\overline{X}-\overline{V})^2 = R^2 \tag{11}$$

Substituting the $\overline{X}$ of Equation (10) into Equation (11) results in

$$\left(\overline{XB} + \overline{WB}\cdot S - \overline{V}\right)^2 - R^2 = 0$$

Let

$$\overline{DX} = \overline{XB} - \overline{V}$$

Therefore,

$$\left(\overline{DX} + \overline{WB}\cdot S\right)^2 - R^2 = 0 \tag{12}$$

Rearranging Equation (12) for a quadratic equation in 'S' gives

$$S^2\left(\overline{WB}^2\right) + 2S\left(\overline{DX}\cdot\overline{WB}\right) + \overline{DX}^2 - R^2 = 0 \tag{13}$$

$\overline{WB}$ $(\overline{W}_x, \overline{W}_y, \overline{W}_z)$ by definition are the direction cosines of the ray.

Thus,

$$\overline{WB}^2 = \overline{W}_x^{\,2} + \overline{W}_y^{\,2} + \overline{W}_z^{\,2} = 1.0$$

Let

B = coefficient of 2S, $(\overline{DX}\cdot\overline{WB})$

C = constant term, $(\overline{DX}^2 - R^2)$

Then Equation (13) becomes

$$S^2 + 2BS + C = 0$$

Solving for S

$$S = -B \pm \sqrt{B^2 - C}$$

If $(B^2-C) < 0$, no real intersection occurs.

If $(B^2-C) = 0$, the ray is tangent to the sphere.

Therefore,

$$\boxed{\text{RIN} = -B - \sqrt{B^2-C}} \tag{14}$$

$$\boxed{\text{ROUT} = -B + \sqrt{B^2-C}} \tag{15}$$

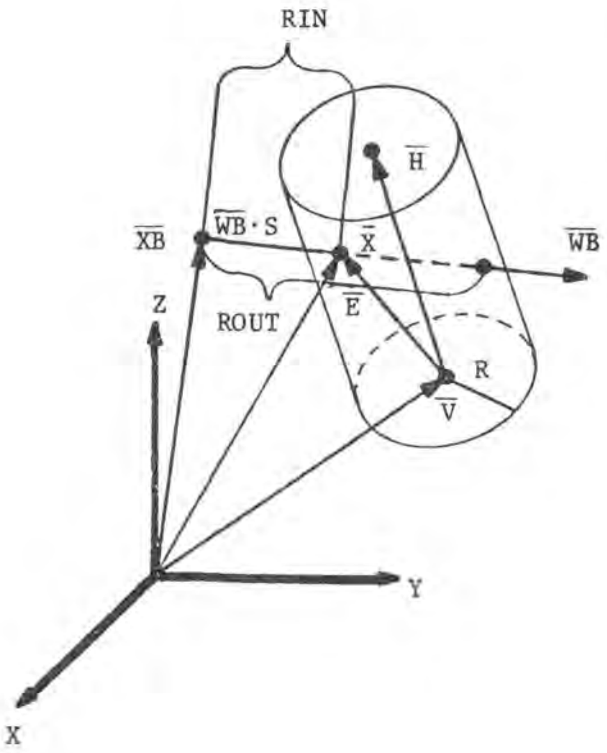
In the case where $(B^2-C) \leq 0$, Subroutine SPH sets RIN = 10^{50} and ROUT = -10^{50} to represent a miss condition.

The constant term $C < 0$ implies that $\overline{DX}^2 < R^2$, which means that the point $\overline{XB}$ (x_o, y_o, z_o) is within the sphere. Therefore, Subroutine SPH sets RIN = -10^{50} and ROUT = $-B + \sqrt{B^2-C}$.

RIGHT CIRCULAR CYLINDER (SUBROUTINE RCC)

Subroutine RCC calculates the intersection distances (if any) of a ray in space and a right circular cylinder. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equations defining the right circular cylinder (see Figure 13).

The right circular cylinder is defined by vertex $\bar{V}$, height vector $\bar{H}$ through the center of the cylinder, and radius R.



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\bar{V}$	vertex of the RCC
$\bar{H}$	height vector of RCC
$\bar{E}$	vector from vertex to point of contact
R	radius of RCC
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 13. Right Circular Cylinder

Any intersect $\bar{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\bar{X} = \overline{XB} + \overline{WB}\cdot S \qquad (16)$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (16) will give a point $\overline{X}$ (x,y,z) along the ray as shown in the figure.

Let h equal the ratio on the $\overline{H}$ vector of the height of the hit on the cylindrical surface as follows

$$h = \frac{\overline{E}\cdot\overline{H}}{\overline{H}\cdot\overline{H}} = \frac{|\overline{E}|\cos\theta}{|\overline{H}|} \tag{17}$$

Therefore the height of the hit with respect to the $\overline{H}$ vector is $h\overline{H}$. Three sides of a right triangle can be defined by the vector, $\overline{E}$, the height of the hit on the $\overline{H}$ vector, $h\overline{H}$, and the radius of the right circular cylinder, R. Therefore

$$\overline{E}\cdot\overline{E} = R^2 + h^2\,\overline{H}\cdot\overline{H} \tag{18}$$

from the Pythagorean theorem.

The vector, $\overline{E}$, can be expressed as

$$\overline{E} = S\cdot\overline{WB} - (\overline{V}-\overline{XB}) \tag{19}$$

Therefore ,

$$\overline{E}\cdot\overline{E} = S^2 + 2S\cdot\overline{WB}\cdot(\overline{XB}-\overline{V}) + (\overline{XB}-\overline{V})\cdot(\overline{XB}-\overline{V}) \tag{20}$$

Also from Equations (17) and (19)

$$h = \frac{(S\cdot\overline{WB} + \overline{XB}-\overline{V})\cdot\overline{H}}{\overline{H}\cdot\overline{H}}$$

and

$$h^2 = \frac{S^2\left(\overline{WB}\cdot\overline{H}\right)^2 + 2S\cdot\overline{WB}\cdot\overline{H}\left(\overline{XB}-\overline{V}\right)\cdot\overline{H} + \left[\left(\overline{XB}-\overline{V}\right)\cdot\overline{H}\right]^2}{\left(\overline{H}\cdot\overline{H}\right)^2} \tag{21}$$

Substituting Equations (20) and (21) into Equation (18) gives:

$$\boxed{\begin{aligned} &S^2\left[1 - \frac{\left(\overline{WB}\cdot\overline{H}\right)^2}{\overline{H}\cdot\overline{H}}\right] - 2S\left[\frac{\overline{WB}\cdot\overline{H}\cdot(\overline{XB}-\overline{V})\cdot\overline{H}}{\overline{H}\cdot\overline{H}} - \overline{WB}\cdot(\overline{XB}-\overline{V})\right] \\ &+ \left[(\overline{XB}-\overline{V})(\overline{XB}-\overline{V}) - \frac{\left[(\overline{XB}-\overline{V})\cdot\overline{H}\right]^2}{\overline{H}\cdot\overline{H}} - R^2\right] = 0 \end{aligned}} \qquad (22)$$

Let

τ equal the coefficient of S^2

λ' equal the coefficient of $2S$

μ' equal the constant term

Therefore Equation (22) becomes

$$\boxed{\tau S^2 - 2\lambda' S + \mu' = 0} \qquad (23)$$

From Equation (22) and (23) $\tau = 0$ if $\overline{WB}$ is parallel to the $\overline{H}$ vector. Therefore no intersection with the side of the cylinder can occur.

For $\lambda = \lambda'/\tau$ and $\mu = \mu'/\tau$ Equation (23) becomes

$$S^2 - 2\lambda S + \mu = 0$$

Solving for S

$$S = \lambda \pm \sqrt{\lambda^2 - \mu}$$

If $(\lambda^2 - \mu) < 0$, no intersection with the cylinder occurs.

If $(\lambda^2 - \mu) = 0$, the ray is tangent to the cylinder.

Therefore,

$$\boxed{RIN = \lambda - \sqrt{\lambda^2 - \mu}} \qquad (24)$$

$$\boxed{ROUT = \lambda + \sqrt{\lambda^2 - \mu}} \qquad (25)$$

for the intersect points with the cylindrical surface.

When an intersect occurs in the plane containing $\overline{V}$ such that vector $\overline{E}$ is within the plane, then vector $\overline{E}$ is perpendicular to the $\overline{H}$ vector. Therefore,

$$\overline{E}\cdot\overline{H} = 0 \tag{26}$$

Substituting Equation (19) into Equation (26) and solving for S (S can equal RIN or ROUT depending on the direction of the ray) results in

$$\boxed{S_V = \frac{(\overline{V}-\overline{XB})\cdot\overline{H}}{\overline{WB}\cdot\overline{H}}} \tag{27}$$

If $S_V \leq 0$ the ray is moving away from the $\overline{V}$ plane and no intersection will occur.

When an intersect occurs in the plane containing $\overline{V}+\overline{H}$ such that the vector $\overline{E}-\overline{H}$ is within the $\overline{V}+\overline{H}$ plane, then vector $\overline{E}-\overline{H}$ is perpendicular to the $\overline{H}$ vector. Therefore

$$(\overline{E}-\overline{H})\cdot\overline{H} = 0 \tag{28}$$

Substititing Equation (19) into Equation (28) and solving for S (S can equal RIN or ROUT depending on the direction of the ray) results in

$$\boxed{S_{V+H} = \frac{\overline{H}\cdot\overline{H} + (\overline{V}-\overline{XB})\cdot\overline{H}}{\overline{WB}\cdot\overline{H}}} \tag{29}$$

If $S_{V+H} \leq 0$, the ray is moving away from the $\overline{V}+\overline{H}$ plane and no intersection will occur.

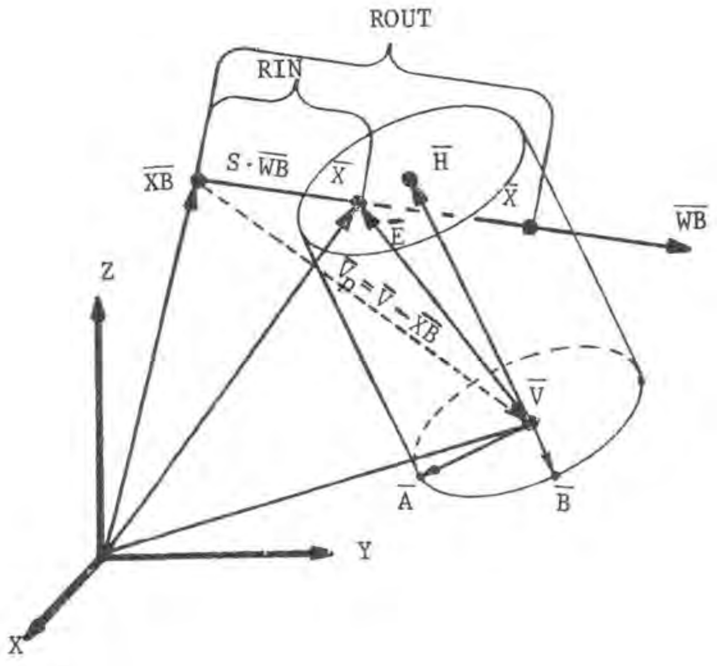
If $\overline{WB}\cdot\overline{H} = 0$, the ray is parallel to the planar surfaces and no intersections with either plane will occur.

If $\overline{WB}\cdot\overline{H} \neq 0$, intersects will occur with the planar surfaces and the cylindrical surface. For a valid intersect on the cylindrical surface it must occur between the two planar surfaces of the RCC. For a valid intersect to be on a planar suface it must occur within the cylindrical cross section of the RCC.

RIGHT ELLIPTIC CYLINDER (SUBROUTINE REC)

Subroutine REC calculates the intersection distances (if any) of a ray in space and a right elliptic cylinder. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equations defining the right elliptic cylinder (see Figure 14).

The right elliptic cylinder is defined by base vertex $\overline{V}$, height vector $\overline{H}$, semi-major axis $\overline{A}$, and semi-minor axis $\overline{B}$.



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	vertex of the REC
$\overline{H}$	height vector of the REC
$\overline{A}$	semi-major axis
$\overline{B}$	semi-minor axis
$\overline{E}$	vector from vertex to point of contact

FIG. 14. Right Elliptic Cylinder

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB}\cdot S \tag{30}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (30) will give a point $\overline{X}$ (x,y,z) along the ray as shown in Figure 14.

The general form of an ellipse which is perpendicular to the Z axis is

$$\boxed{\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1} \tag{31}$$

or
$$b^2x^2 + a^2y^2 = a^2b^2 \tag{32}$$

where $a^2 = \overline{A} \cdot \overline{A}$ and $b^2 = \overline{B} \cdot \overline{B}$

The origin of the ray $\overline{XB}$ and the direction cosines of the ray $\overline{WB}$ can be transformed into the coordinate system of the REC, where $\overline{V}$ now becomes the origin, the x-axis is along the semi-major axis, the y-axis is along the semi-minor axis, and the z-axis is coincident with the $\overline{H}$ vector, by the dot products

$$\overline{VP}_A = (\overline{V} - \overline{XB}) \cdot \overline{A}$$
$$\overline{VP}_B = (\overline{V} - \overline{XB}) \cdot \overline{B}$$
$$\overline{W}_A = \overline{WB} \cdot \overline{A}$$
$$\overline{W}_B = \overline{WB} \cdot \overline{B}$$

The intersection of the elliptic cylinder of Equation (32) and the transformed ray from the above dot products can now be considered, where the new intersect equation is

$$\boxed{\overline{E} = \overline{VP} + S \cdot \overline{W}} \tag{33}$$

which on substituting into Equation (32) yields

$$b^2(\overline{VP}_A + \overline{W}_A \cdot S)^2 + a^2(\overline{VP}_B + \overline{W}_B \cdot S)^2 = a^2b^2 \tag{34}$$

or

$$\left(\overline{B}\cdot\overline{B}\right)\left(\overline{VP}_A + \overline{W}_A\cdot S\right)^2 + \left(\overline{A}\cdot\overline{A}\right)\left(\overline{VP}_B + \overline{W}_B\cdot S\right)^2 = \left(\overline{A}\cdot\overline{A}\right)\left(\overline{B}\cdot\overline{B}\right)$$

Collecting terms and regrouping for a quadratic equation in S results in

$$\boxed{\begin{aligned} &S^2\left[\left(\overline{B}\cdot\overline{B}\right)\left(\overline{W}_A\right)^2 + \left(\overline{A}\cdot\overline{A}\right)\left(\overline{W}_B\right)^2\right] \\ &-2S\left[\left(\overline{B}\cdot\overline{B}\right)\left(\overline{VP}_A\cdot\overline{W}_A\right) + \left(\overline{A}\cdot\overline{A}\right)\left(\overline{VP}_B\cdot\overline{WB}\right)\right] \\ &+\left[\left(\overline{B}\cdot\overline{B}\right)\left(\overline{VP}_A\right)^2 + \left(\overline{A}\cdot\overline{A}\right)\left(\overline{VP}_B\right)^2 - \left(\overline{A}\cdot\overline{A}\right)\left(\overline{B}\cdot\overline{B}\right)\right] = 0\end{aligned}} \tag{35}$$

Let

τ equal the coefficient of S^2

λ' equal the coefficient of 2S

μ' equal the constant term

Therefore Equation (35) becomes

$$\tau S^2 - 2\lambda' S + \mu' = 0 \tag{36}$$

For $\lambda = \lambda'/\tau$ and $\mu = \mu'/\tau$ Equation (36) becomes

$$S^2 - 2\lambda S + \mu = 0$$

Solving for S

$$S = \lambda \pm \sqrt{\lambda^2-\mu}$$

If $(\lambda^2-\mu) < 0$, no intersection with the elliptic cylinder occurs.

If $(\lambda^2-\mu) = 0$, the ray is tangent to the cylinder.

If the coefficient of S^2, τ, is zero, and since from Equation (35) $\overline{W}_A = \overline{WB}\cdot\overline{A} = 0$ and $\overline{W}_R = \overline{WB}\cdot\overline{B} = 0$ when the ray is parallel to the $\overline{H}$ vector (or perpendicular to the $\overline{A}$ and $\overline{B}$ vectors) intersections can only occur with the planar surfaces.

Therefore, possible candidates for RIN and ROUT with the quadratic surface are

$$\boxed{RIN = \lambda - \sqrt{\lambda^2 - \mu}} \tag{37}$$

$$\boxed{ROUT = \lambda + \sqrt{\lambda^2 - \mu}} \tag{38}$$

When an intersect occurs in the plane containing $\overline{V}$

$$\boxed{S_V = \overline{VP}_H / \overline{W}_H} \tag{39}$$

and for an intersect occurring in the $\overline{V}+\overline{H}$ plane

$$\boxed{S_{V+H} = S_V + (\overline{H} \cdot \overline{H}) / \overline{W}_H} \tag{40}$$

Therefore, RIN = S_V and ROUT = S_{V+H} when $\overline{W}_H > 0$, or RIN = S_{V+H} and ROUT = S_V when $\overline{W}_H < 0$. When $\overline{W}_H = 0$, the ray is parallel to the two planes, and intersections can occur only with the quadratic surface.

When RIN or ROUT is an intersect with the quadratic surface the intersect must lie between the two planar surfaces to be a valid intersect. Therefore,

$$\boxed{0 \leq \left(S \cdot \overline{W}_H - \overline{VP}_H\right) \leq \left(\overline{H} \cdot \overline{H}\right)} \tag{41}$$

where S is either RIN or ROUT for the quadratic surface.

When RIN or ROUT is an intersect with a planar surface the intersect must lie within the elliptic cross section to be a valid intersect. Therefore, the evaluation of Equation (36) must be negative. Therefore,

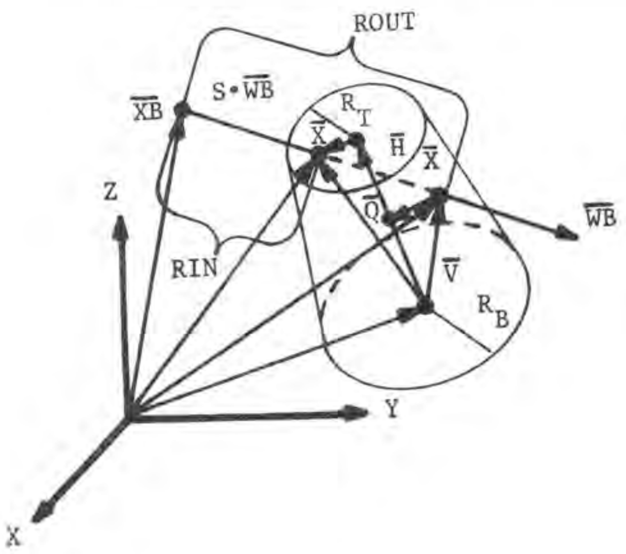
$$\boxed{S^2 - 2\lambda S + \mu \leq 0} \tag{42}$$

where S is the RIN or ROUT value.

RIGHT TRUNCATED CONE (SUBROUTINE TRC)

Subroutine TRC calculates the intersection distances (if any) of a ray in space and a right truncated cone. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equations defining the right truncated cone.

The right truncated cone is defined by a vertex, $\overline{V}$, a height vector, $\overline{H}$, a radius at the vertex, R_B, and a radius at $\overline{V}+\overline{H}$ R_T (see Figure 15).



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{H}$	height vector of TRC
$\overline{V}$	vertex of the TRC
R_B	radius of the base of the TRC
R_T	radius of the top of the TRC
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 15. Right Truncated Cone

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB}\cdot S \tag{43}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (43) will give a point $\overline{X}$ (x,y,z) along the ray as shown in the figure.

Let $\overline{Q}$ be a point along the height vector such that the vector $(\overline{Q}-\overline{X})$ is perpendicular to the height vector $\overline{H}$, or

$$(\overline{Q}-\overline{X})\cdot\overline{H} = 0 \tag{44}$$

$\overline{Q}$ can then be expressed as

$$\overline{Q} = \overline{V} + \gamma\overline{H} \tag{45}$$

where $0 \leq \gamma \leq 1.0$, and $\gamma\overline{H}$ is the distance from $\overline{V}$ to $\overline{Q}$. Substituting Equations (43) and (45) into Equation (44) results in

$$(\overline{V}+\gamma\overline{H}-\overline{XB}-\overline{WB}\cdot S)\cdot\overline{H} = 0$$

Solving for γ results in

$$\gamma = \frac{(\overline{XB}-\overline{V})\cdot\overline{H} + \overline{WB}\cdot\overline{H}\cdot S}{\overline{H}\cdot\overline{H}} \tag{46}$$

The radius of the TRC at point $\overline{Q}$ is expressed by

$$R = (R_T-R_B)\ \gamma + R_B \tag{47}$$

Three sides of a right triangle can be defined by the height of the hit on the $\overline{H}$ vector, $\gamma\overline{H}$, the radius of the TRC at $\overline{Q}$, R, and the vector between $\overline{V}$ and $\overline{X}$. Therefore,

$$(\overline{V}-\overline{X})^2 = (\gamma\overline{H})^2 + R^2 \tag{48}$$

from the Pythagorean theorem. Substituting Equations (43), (46), and (47) into Equation (48) and solving for S, the following quadratic equation in S is obtained.

$$\boxed{\begin{aligned} & S^2 \left\{ 1 - \frac{(\overline{WB}\cdot\overline{H})^2 \left[1.0 + (R_T-R_B)^2 \right]}{\overline{H}\cdot\overline{H}} \right\} \\ & -2S \left\{ -(\overline{XB}-\overline{V})\cdot\overline{WB} + \frac{(\overline{WB}\cdot\overline{H})\cdot\left[(\overline{XB}-\overline{V})\cdot\overline{H} + C_1\right]}{\overline{H}\cdot\overline{H}} \right\} \\ & + \left\{ \left[(\overline{XB}-\overline{V})^2 - C_2{}^2\right] - \frac{\left[(XB-V)\cdot H\right]^2}{\overline{H}\cdot\overline{H}} \right\} = 0 \end{aligned}} \tag{49}$$

where

$$C_1 = \left(R_T - R_B\right)\left\{R_B - \frac{\left(R_T - R_B\right)\left[\left(\overline{V} - \overline{XB}\right)\cdot\overline{H}\right]}{\overline{H}\cdot\overline{H}}\right\}$$

$$C_2 = R_B - \frac{\left(R_T - R_B\right)\left[\left(\overline{V} - \overline{XB}\right)\cdot\overline{H}\right]}{\overline{H}\cdot\overline{H}}$$

Let

τ equal the coefficient of S^2

λ' equal the coefficient of 2S

μ' equal the constant term

Therefore Equation (49) becomes

$$\tau S^2 - 2\lambda' S + \mu' = 0 \tag{50}$$

From Equations (49) and (50) $\tau = 0$ only when $\overline{WB}$ is parallel to the $\overline{H}$ vector and $R_T = R_B$. For those conditions no intersections with the side of the cylinder could take place.

For $\lambda = \lambda'/\tau$ and $\mu = \mu'/\tau$ Equation (50) becomes

$$S^2 - 2\lambda S + \mu = 0$$

Solving for S

$$S = \lambda \pm \sqrt{\lambda^2 - \mu}$$

If $(\lambda^2 - \mu) < 0$, no intersection with the TRC occurs.

If $(\lambda^2 - \mu) = 0$, the ray is tangent to the TRC.

If $(\lambda^2 - \mu) > 0$, then:

$$\boxed{RIN = \lambda - \sqrt{\lambda^2 - \mu}} \tag{51}$$

$$\boxed{ROUT = \lambda + \sqrt{\lambda^2 - \mu}} \tag{52}$$

are the possible intersect points with the quadratic surface.

From Equation (45), γ must satisfy $0 \leq \gamma \geq 1$. Substituting Equation (46) for γ results in

$$\boxed{0 \leq (\overline{XB}-\overline{V})\cdot\overline{H} + \overline{WB}\cdot\overline{H}\cdot S \leq \overline{H}\cdot\overline{H}} \qquad (53)$$

for S to intersect the real part of the cone, where S can equal either RIN or ROUT.

When an intersect occurs in the $\overline{V}$ plane such that the vector $(\overline{X}-\overline{V})$ lies within the plane, the vector $(\overline{X}-\overline{V})$ is perpendicular to the $\overline{H}$ vector. Therefore,

$$(\overline{X}-\overline{V})\cdot\overline{H} = 0 \qquad (54)$$

From the figure it can be seen that

$$(\overline{X}-\overline{V}) = \overline{WB}\cdot S - (\overline{V}-\overline{XB}) \qquad (55)$$

Substituting Equation (55) into Equation (54) and solving for S (S can equal RIN or ROUT depending on the direction of the ray) results in

$$S_V = \frac{(\overline{V}-\overline{XB})\cdot\overline{H}}{\overline{WB}\cdot\overline{H}} \qquad (56)$$

If $S_V \leq 0$, the ray is moving away from the $\overline{V}$ plane, and no intersection will occur.

For an intersect in the $\overline{V}$ plane to be valid, the length of the vector from $\overline{V}$ to the intersect $\overline{X}$, or $(\overline{X}-\overline{V})$, must be equal to or less than the radius of the base plane, R_B, or

$$\text{Length of } (\overline{X}-\overline{V}) \leq \text{length of } R_B$$

and

$$(\overline{X}-\overline{V})\cdot(\overline{X}-\overline{V}) \leq R_B\cdot R_B$$

From Equation (55)

$$(\overline{X}-\overline{V}) = S\cdot\overline{WB} - (\overline{V}-\overline{XB})$$

or

$$[S\cdot\overline{WB}-(\overline{V}-\overline{XB})]\,[S\cdot\overline{WB}-(\overline{V}-\overline{XB})] \le R_B^{\,2}$$

which results in

$$\boxed{S^2 - 2S\cdot[\overline{WB}\cdot(\overline{V}-\overline{XB})] + (\overline{V}-\overline{XB})^2 - R_B^{\,2} \le 0} \tag{57}$$

Therefore, for a valid intersect in the $\overline{V}$ plane to occur, Equation(57) must be satisfied for S equal to S_V.

When an intersect occurs in the $\overline{V}+\overline{H}$ plane such that the vector $[\overline{X}-(\overline{V}+\overline{H})]$ lies within the plane, the vector $[\overline{X}-(\overline{V}+\overline{H})]$ is perpendicular to the $\overline{H}$ vector. Therefore,

$$[(\overline{X}-\overline{V})-\overline{H}]\cdot\overline{H} = 0 \tag{58}$$

Substituting Equation(55) into Equation(54) and solving for S (S can equal RIN or ROUT depending on the direction of the ray) results in

$$\boxed{S_{V+H} = \frac{(\overline{V}-\overline{XB})\cdot\overline{H} + \overline{H}\cdot\overline{H}}{\overline{WB}\cdot\overline{H}}} \tag{59}$$

If $S_{V+H} \le 0$, the ray is moving away from the $\overline{V}+\overline{H}$ plane, and no intersection occurs.

For an intersect in the $\overline{V}+\overline{H}$ plane to be valid, the length of the vector from $\overline{V}+\overline{H}$ to the intersect X, or $[\overline{X}-(\overline{V}+\overline{H})]$, must be equal to or less than the radius of the top plane, R_T, or

$$\text{Length of } [(\overline{X}-\overline{V})-\overline{H}] \le \text{length of } R_T$$

and

$$[(\overline{X}-\overline{V})-\overline{H}]\cdot[(\overline{X}-\overline{V})-\overline{H}] \le R_T\cdot R_T$$

From Equation (55)

$$(\overline{X}-\overline{V}) = S\cdot\overline{WB} - (\overline{V}-\overline{XB})$$

or

$$[S\cdot\overline{WB}-(\overline{V}-\overline{XB})-\overline{H}]\ [S\cdot\overline{WB}-(\overline{V}-\overline{XB})-\overline{H}] \le {R_T}^2$$

which results in

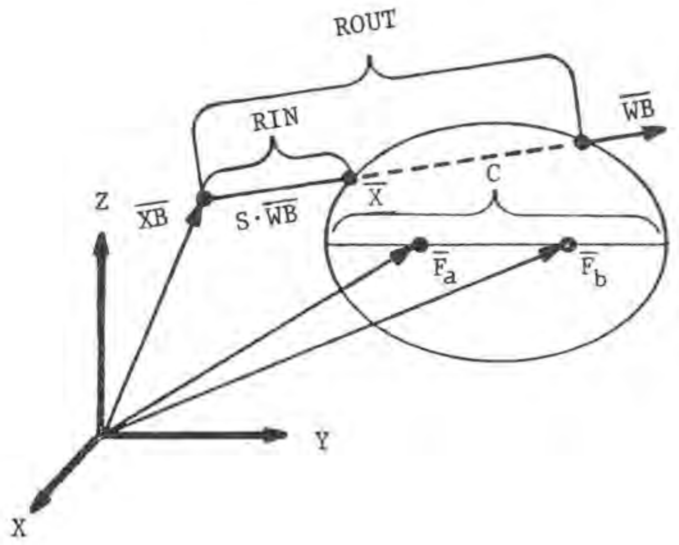
$$\boxed{S^2-2S\cdot\left[\overline{WB}\cdot(\overline{V}-\overline{XB}) + (\overline{WB}\cdot\overline{H})\right] + (\overline{V}-\overline{XB})^2 + 2(\overline{V}-\overline{XB})\cdot\overline{H} + \overline{H}\cdot\overline{H} - {R_T}^2 \le 0} \qquad (60)$$

Therefore, for a valid intersect in the $\overline{V}+\overline{H}$ plane to occur, Equation (60) must be satisfied for S equal to S_{V+H}.

If $\overline{WB}\cdot\overline{H} = 0$ the ray is parallel to the planar surfaces, and no intersections with either plane will occur.

ELLIPSOID OF REVOLUTION (SUBROUTINE ELL)

Subroutine ELL calculates the intersection distances (if any) of a ray in space and an ellipsoid of revolution. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equation defining the ellipsoid of revolution (see Figure 16).



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
C	length of the major axis
$\overline{F}_a,\overline{F}_b$	foci of the ellipsoid
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 16. Ellipsoid of Revolution

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB}\cdot S \tag{61}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (61) will give a point $\overline{X}$ (x,y,z) along the ray.

For any point $\overline{X}$ on the surface of an ellipsoid with foci $\overline{F}_a$ and $\overline{F}_b$ and major axis length C, the equation is

$$\sqrt{(\overline{X}-\overline{F}_a)^2} + \sqrt{(\overline{X}-\overline{F}_b)^2} = C \tag{62}$$

where the terms $\sqrt{(\overline{X}-\overline{F}_a)^2}$ and $\sqrt{(\overline{X}-\overline{F}_b)^2}$ are the distance from point $\overline{X}$ on the surface of the ellipsoid to the foci.

Substituting Equation (61) into Equation (62) results in

$$\sqrt{\left(\overline{XB} + \overline{WB}\cdot S - \overline{F}_a\right)^2 + \left(\overline{XB} + \overline{WB}\cdot S - \overline{F}_b\right)^2} = C$$

Let

$$\overline{D}_1 = \overline{XB} - \overline{F}_a$$

$$\overline{D}_2 = \overline{XB} - \overline{F}_b$$

Therefore

$$\sqrt{\left(\overline{D}_1 + \overline{WB}\cdot S\right)^2} + \sqrt{\left(\overline{D}_2 + \overline{WB}\cdot S\right)} = C$$

or

$$\sqrt{\left(\overline{D}_1 + \overline{WB}\cdot S\right)^2} = C - \sqrt{\left(\overline{D}_2 + \overline{WB}\cdot S\right)^2} \tag{63}$$

Squaring both sides of Equation (63) and collecting term gives

$$\boxed{\frac{\overline{D}_1^{\,2} - \overline{D}_2^{\,2} - C^2}{-2C} + \frac{\left(2\overline{D}_1 \cdot \overline{WB} - 2\overline{D}_2 \cdot \overline{WB}\right)}{-2C}\quad S = \sqrt{\left(\overline{D}_2 + \overline{WB}\cdot\overline{S}\right)^2}} \tag{64}$$

Letting

$$A = \frac{\overline{D}_1^{\,2} - \overline{D}_2^{\,2} - C^2}{-2C} \qquad \text{(the first term)}$$

and

$$B = \frac{2\overline{D}_1 \cdot \overline{WB} - 2\overline{D}_2 \cdot \overline{WB}}{-2C} \qquad \text{(the second term)}$$

gives

$$A + B \cdot S = \sqrt{\left(\overline{D}_2 + \overline{WB} \cdot S\right)^2} \tag{65}$$

Squaring Equation (65), collecting terms for a quadratic in S, and noting that $\overline{WB}$ $(\overline{W}_x, \overline{W}_y, \overline{W}_z)$ by definition are the direction cosines of the ray and $\overline{WB}^2 = \overline{W}_x^{\,2} + \overline{W}_y^{\,2} + \overline{W}_z^{\,2} = 1.0$ gives

$$S^2\left(B^2-1\right) + 2S\left(A \cdot B - \overline{D}_2 \cdot \overline{WB}\right) + \left(A^2 - \overline{D}_2^{\,2}\right) = 0 \tag{66}$$

Letting

τ = coefficient of S^2

λ' = coefficient of 2S

μ' = constant term

gives

$$\tau S^2 + 2\lambda' S + \mu' = 0$$

If $\lambda = \lambda'/\tau$ and $\mu = \mu'/\tau$ then $S^2 + 2\lambda S + \mu = 0$ and

$$\boxed{\lambda = \frac{A \cdot B - \overline{D}_2^{\,2} \cdot \overline{WB}}{B^2-1}} \tag{67}$$

$$\boxed{\mu = \frac{A^2 - \overline{D}_2^{\,2}}{B^2 - 1}} \qquad (68)$$

Solving for S

$$S = -\lambda \pm \sqrt{\lambda^2 - \mu}$$

If $(\lambda^2 - \mu) < 0$, no real intersection occurs.

If $(\lambda^2 - \mu) = 0$, the ray is tangent to the ellipsoid.

Therefore,

$$\boxed{RIN = -\lambda - \sqrt{\lambda^2 - \mu}} \qquad (69)$$

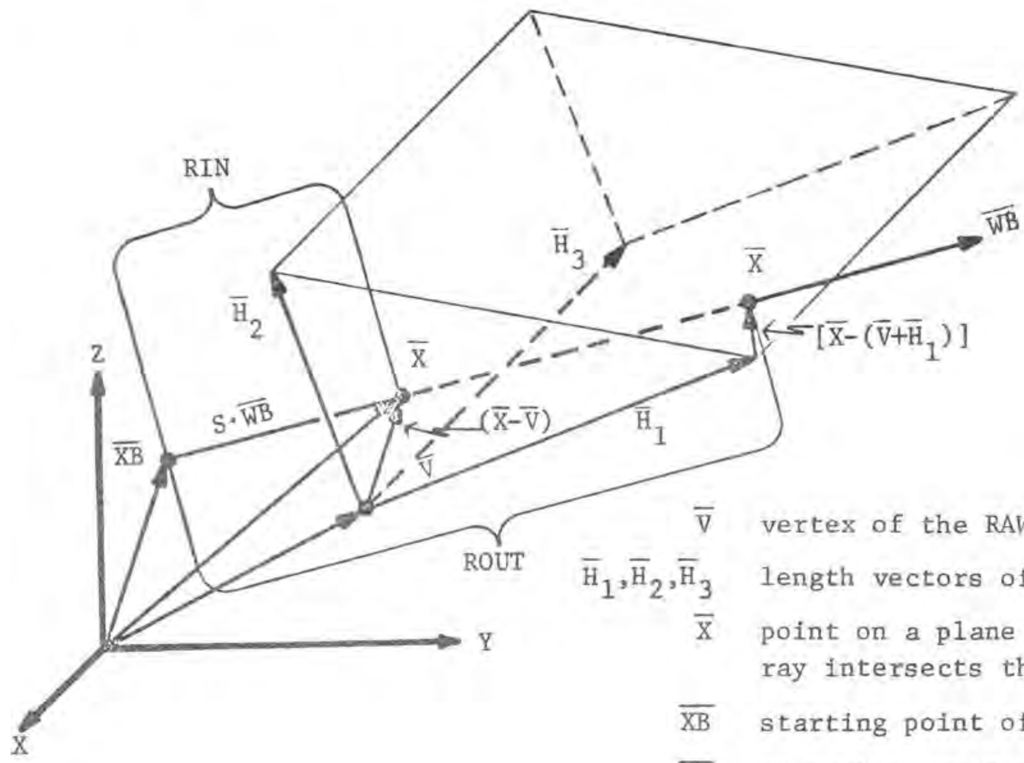
$$\boxed{ROUT = -\lambda + \sqrt{\lambda^2 - \mu}} \qquad (70)$$

In the case where $(\lambda^2 - \mu) \leq 0$ Subroutine ELL sets RIN = 10^{50} and ROUT = -10^{50} to represent a miss condition.

RIGHT ANGLE WEDGE (SUBROUTINE RAW)

Subroutine RAW calculates the intersection distances (if any) of a ray in space and a right angle wedge. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equations defining the sides of the right angle wedge.

The right angle wedge is defined by vertex $\overline{V}$ and length vectors $\overline{H}_1$, $\overline{H}_2$, and $\overline{H}_3$. The right angle wedge has five sides; the diagonal side is always opposite the $\overline{H}_3$ length vector as shown in Figure 17.



$\overline{V}$	vertex of the RAW
$\overline{H}_1,\overline{H}_2,\overline{H}_3$	length vectors of the RAW
$\overline{X}$	point on a plane where the ray intersects the plane
$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$S\cdot\overline{WB}$	distance from $\overline{XB}$ to point where ray intersects plane
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 17. Right Angle Wedge

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB} \cdot S \tag{71}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (71) will give a point $\overline{X}$ (x,y,z) along the ray.

Each of the five planes of the right angle wedge can be expressed in terms of any vector within the plane, and the normal to the plane, since the dot product of the two is equal to zero.

Therefore, the dot product of a vector from any point $\overline{X}$ in the plane to the vertex (or vertex plus height vector) and the length vector $\overline{H}$ perpendicular to the side containing $\overline{X}$ can be expressed for the five sides as follows:

For the rectangular side perpendicular to $\overline{H}_1$

$$(\overline{X}-\overline{V}) \cdot \overline{H}_1 = 0$$

For the rectangular side perpendicular to $\overline{H}_2$

$$(\overline{X}-\overline{V}) \cdot \overline{H}_2 = 0$$

For the triangular side containing $\overline{V}$

$$(\overline{X}-\overline{V}) \cdot \overline{H}_3 = 0$$

For the triangular side containing $\overline{V}+\overline{H}_3$

$$[\overline{X} - (\overline{V}+\overline{H}_3)] \cdot \overline{H}_3 = 0$$

For the slanted side opposite $\overline{H}_3$

$$[\overline{X} - (\overline{V}+\overline{H}_1)] \cdot (\overline{H}_1^{\,2} \cdot \overline{H}_2 + \overline{H}_2^{\,2} \cdot \overline{H}_1) = 0$$

The term $(\overline{H}_1{}^2 \cdot \overline{H}_2 + \overline{H}_2{}^2 \cdot \overline{H}_1)$ is the normal to the slanted side and is derived as follows:

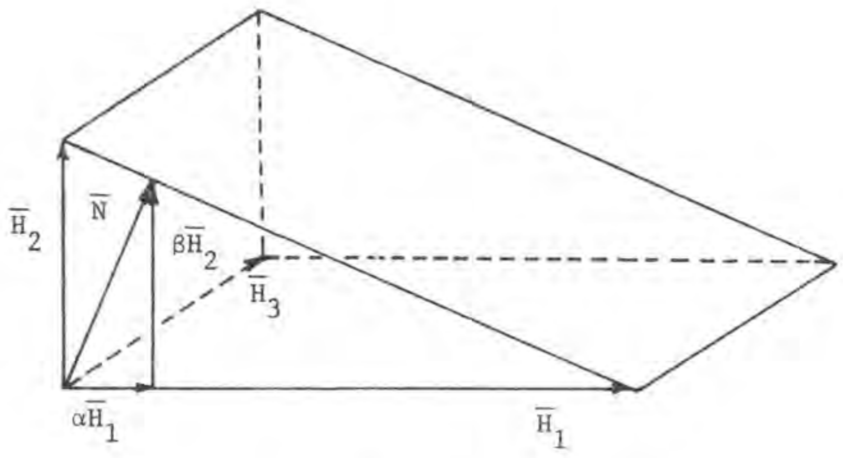


FIG. 18. Right Angle Wedge Normal Derivation

From Figure 18, $\overline{N}$ is perpendicular to the slanted edge and is equivalent to

$$\overline{N} = \alpha\overline{H}_1 + \beta\overline{H}_2$$

where

α and β are proportionality constants

Let

H_1 and H_2 equal the magnitudes of vectors $\overline{H}_1$ and $\overline{H}_2$

respectively.

Therefore, by similar triangles

$$\frac{\alpha H_1}{\beta H_2} = \frac{H_2}{H_1} \text{ and } \alpha = \beta \frac{H_2{}^2}{H_1{}^2}$$

Therefore,

$$\overline{N} = \beta \frac{H_2{}^2}{H_1{}^2} \cdot \overline{H}_1 + \beta \cdot \overline{H}_2$$

and

$$\frac{H_1{}^2}{\beta} \cdot \overline{N} = H_2{}^2 \cdot \overline{H}_1 + H_1{}^2 \cdot \overline{H}_2$$

Since $\frac{H_1{}^2}{\beta}$ is a scaler which affects the length of $\overline{N}$, it can be ignored.

Therefore $\overline{N} = \overline{H}_2{}^2 \cdot \overline{H}_1 + \overline{H}_1{}^2 \cdot H_2$ since $\overline{H}^2 = \overline{H} \cdot \overline{H} = H^2$

Substituting ray Equation (71) into each of the five side equations and solving for S, the distance from the ray origin, $\overline{XB}$, to each side, results in

For the rectangular side perpendicular to $\overline{H}_1$

$$S_3 = \frac{(\overline{V}-\overline{XB}) \cdot \overline{H}_1}{\overline{WB} \cdot \overline{H}_1}$$

For the rectangular side perpendicular to $\overline{H}_2$

$$S_1 = \frac{(\overline{V}-\overline{XB}) \cdot \overline{H}_2}{\overline{WB} \cdot \overline{H}_2}$$

For the triangular side containing $\overline{V}$

$$S_5 = \frac{(\overline{V}-\overline{XB}) \cdot \overline{H}_3}{\overline{WB} \cdot \overline{H}_3}$$

For the triangular side containing $\overline{V}+\overline{H}_3$

$$S_6 = \frac{(\overline{V}-\overline{XB}) \cdot \overline{H}_3 + (\overline{H}_3)^2}{\overline{WB} \cdot \overline{H}_3}$$

For the slanted side opposite $\overline{H}_3$

$$S_2 = \frac{(\overline{V}-\overline{XB}) \cdot \overline{N} + (\overline{H}_1 \cdot \overline{N})}{\overline{WB} \cdot \overline{N}}$$

and $$S_2 = \frac{\left(\overline{H}_1 \cdot \overline{H}_1\right)\left(\overline{H}_2^2\right) - \left[(\overline{XB}-\overline{V}) \cdot \overline{H}_1\right]\left(\overline{H}_2^2\right) - \left[(\overline{XB}-\overline{V}) \cdot \overline{H}_2\right]\left(\overline{H}_1^2\right)}{\left(\overline{H}_2^2\right)\left(\overline{WB} \cdot \overline{H}_1\right) + \left(\overline{H}_1^2\right)\left(\overline{WB} \cdot \overline{H}_2\right)}$$

since $\overline{N} = \overline{H}_1^2 \cdot \overline{H}_2 + \overline{H}_2^2 \cdot \overline{H}_1$ and $\overline{H}_1 \cdot \overline{H}_2 = 0$

where

$$\overline{N} = \overline{H}_1^{\,2} \cdot \overline{H}_2 + \overline{H}_2^{\,2} \cdot \overline{H}_1$$

In the above distance equations if the value of the numerator and the denominator are such that the value of the distance, S, would be negative, the ray is moving away from the RAW and no intersection will occur.

If the ray hits the RAW, there will be two intersects. RIN will be the minimum positive S, and ROUT will be the maximum positive S.

In order to simplify computation let

$$\boxed{A_i = \overline{H}_i \cdot \overline{H}_i} \tag{72}$$

$$\boxed{P_i = (\overline{XB} - \overline{V}) \cdot \overline{H}_i} \tag{73}$$

$$\boxed{G_i = \overline{WB} \cdot \overline{H}_i} \tag{74}$$

where

$i = 1, 2, 3$

Therefore, the distance equations become

$$\boxed{S_3 = -P_1/G_1} \tag{75}$$

$$\boxed{S_1 = -P_2/G_2} \tag{76}$$

$$\boxed{S_5 = -P_3/G_3} \tag{77}$$

$$\boxed{S_6 = (-P_3 + A_3)/G_3} \tag{78}$$

substituting Equations (72), (73), and (74) into the expression for S_2 from the previous page results in

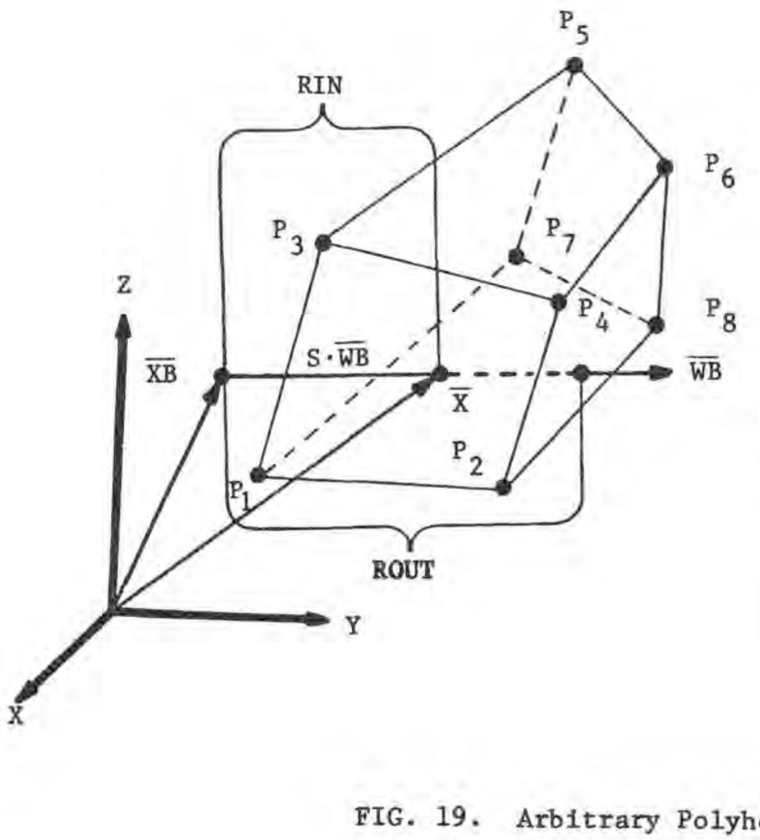
$$\boxed{S_2 = (A_1 \cdot A_2 - P_1 A_2 - P_2 A_1)/(A_2 G_1 + A_1 G_2)} \tag{79}$$

ARBITRARY POLYHEDRON (SUBROUTINE ARB)

Subroutine ARB calculates the intersection distances (if any) of a ray in space and an arbitrary polyhedron. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equations defining the six planes of the arbitrary polyhedron.

The arbitrary polyhedron is described by eight points which define the six planar surfaces shown in Figure 19 below:

$$\begin{aligned}
S_1 &= P_1, P_2, P_4, P_3\\
S_2 &= P_7, P_8, P_6, P_5\\
S_3 &= P_1, P_2, P_8, P_7\\
S_4 &= P_3, P_4, P_6, P_5\\
S_5 &= P_1, P_3, P_5, P_7\\
S_6 &= P_2, P_4, P_6, P_8
\end{aligned}$$



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$S \cdot \overline{WB}$	distance from $\overline{XB}$ to point where ray intersects plane
$\overline{X}$	point on plane where ray intersects plane
P_1-P_8	eight points which describe the arbitrary polyhedron
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 19. Arbitrary Polyhedron

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB} \cdot \overline{S} \tag{80}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus for any value of scalar S, Equation (80) will give a point $\overline{X}$ (x,y,z) along the ray.

Every plane is represented by an equation of the first degree in one of the variables x,y,z. The general equation of a plane is

$$Ax + By + Cz + D = 0$$

provided that A, B, and C are not all zero. A plane can be defined as passing through three points (three-point form) $P_1 = (x_1, y_1, z_1)$, $P_2 = (x_2, y_2, z_2)$, and $P_3 = (x_3, y_3, z_3)$, which must not lie in a straight line, by expanding the determinant

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0$$

or,

$$\begin{vmatrix} y_1 & z_1 & 1 \\ y_2 & z_2 & 1 \\ y_3 & z_3 & 1 \end{vmatrix} x + \begin{vmatrix} z_1 & x_1 & 1 \\ z_2 & x_2 & 1 \\ z_3 & x_3 & 1 \end{vmatrix} y + \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} z - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} = 0$$

therefore, from the general form of the equation

$$A = y_1z_3 - y_1z_2 + y_2z_1 - y_2z_3 + y_3z_2 - y_3z_1$$

$$B = x_1z_2 - x_1z_3 + x_2z_3 - x_2z_1 + x_3z_1 - x_3z_2$$

$$C = x_1y_3 - x_1y_2 + x_2y_1 - x_2y_3 + x_3y_2 - x_3y_1$$

$$D = x_1(y_2z_3 - z_2y_3) - x_2(y_1z_3 - z_1y_3) + x_3(y_1z_2 - z_1y_2)$$

Each of the six planes of the arbitrary polyhedron is described by the equation of a plane (after the eight points are converted into six plane equations) where the sign of the coefficients is adjusted such that

$$\boxed{A_iX + B_iY + C_iZ + D_i \geq 0} \qquad (81)$$

for all (x,y,z) on the surface or interior of the figure where i = 1-6 for each of the six sides.

The direction cosines of a normal to the defined plane are given by

$$\overline{WB}_x = \frac{A}{\sqrt{A^2+B^2+C^2}}$$

$$\overline{WB}_y = \frac{B}{\sqrt{A^2+B^2+C^2}}$$

$$\overline{WB}_z = \frac{C}{\sqrt{A^2+B^2+C^2}}$$

Substituting the components of ray Equation (80) into the equation for the six sides of the ARB results in

$$A_i\cdot(\overline{XB}_x + S\cdot\overline{WB}_x) + B_i\cdot(\overline{XB}_y + S\cdot\overline{WB}_y) + C_i\cdot(\overline{XB}_z + S\cdot\overline{WB}_z) + D_i = 0$$

Solving for S

$$\boxed{S_i = \frac{-\left(A_i\cdot\overline{XB}_x + B_i\cdot\overline{XB}_y + C_i\cdot\overline{XB}_z + D_i\right)}{A_i\cdot\overline{WB}_x + B_i\cdot\overline{WB}_y + C_i\cdot\overline{WB}_z}} \qquad (82)$$

Therefore, given S_i, the point of intersection with each plane is given by

$$\boxed{\overline{XI}_i = \overline{XB} + S_i\cdot\overline{WB}} \qquad (83)$$

$\overline{XI}_i$ lies on the surface of the arbitrary polyhedron if

$$\boxed{A_jX + B_jY + C_jZ + D_j \geq 0} \qquad (84)$$

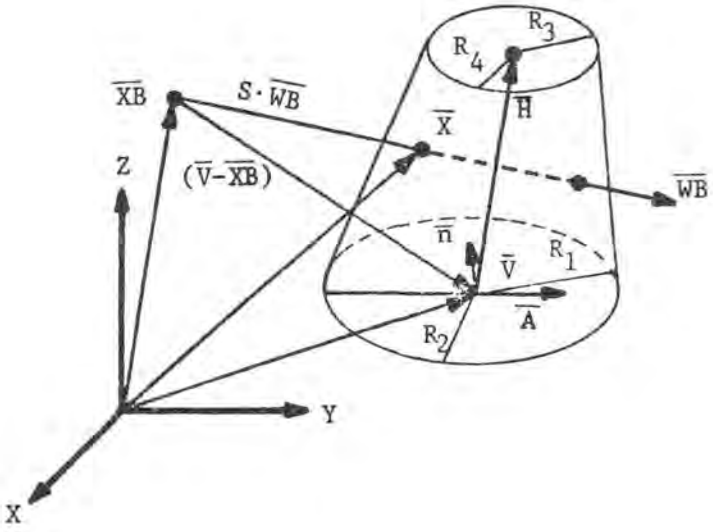
where X, Y, Z are the components of the $\overline{XI}_i$ and j = 1-6 for each of the six sides.

Since the ray can only intersect two sides of the ARB, Equation (84) will be satisfied for two planes only. Therefore, RIN is the minimum S_i, and ROUT is the maximum S_i. If ROUT < 0 or if none of the six S_i's satisfies Equation (84), no intersection occurs with the arbitrary polyhedron.

TRUNCATED ELLIPTIC CONE (SUBROUTINE TEC)

Subroutine TEC calculates the intersection distances (if any) of a ray in space and a truncated elliptic cone. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equation that defines the truncated elliptic cone.

The truncated elliptic cone is defined by a vertex, $\overline{V}$; a height vector, $\overline{H}$; a unit vector, $\overline{N}$, normal to the base ellipse; the radius along the semi-major axis of the base ellipse, R1; the radius along the semi-minor axis of the base ellipse, R2; the ratio of the base ellipse to the top ellipse, RR, from which is computed the radius along the semi-major axis of the top ellipse, R3; and the radius along the semi-minor axis of the top ellipse, R4.



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	vertex of the TEC
$\overline{H}$	height vector of the TEC
$\overline{N}$	normal unit vector to the base ellipse
$\overline{A}$	semi-major axis of base ellipse (unit vector)
R1	radius along the semi-major axis of base ellipse
R2	radius along the semi-minor axis of base ellipse
R3	radius along the semi-major axis of top ellipse
R4	radius along the semi-minor axis of top ellipse

FIG. 20 Truncated Elliptic Cone

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{S}\cdot\overline{WB} \tag{85}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (85) will give a point $\overline{X}(x,y,z)$ along the ray as shown in Figure 20.

Case 1

Compute the intersect distances (if any), RIN and ROUT, for a ray that is parallel to the top and base planes where

$$\overline{WB}\cdot\overline{N} = 0$$

Since the normal, $\overline{N}$, is a unit vector, the distance from the base ellipse to the top ellipse is the projection of the height vector, $\overline{H}$, onto the normal, $\overline{N}$, and is represented as

$$\overline{H}\cdot\overline{N}$$

The projection of the $(\overline{V}-\overline{XB})$ vector onto the normal, which is the distance from the base ellipse to the height of the intersect for a ray parallel to the planar surfaces, is represented as

$$-(\overline{V}-\overline{XB})\cdot\overline{N}$$

Therefore, the ratio, γ, of the height of the intersect ellipse to the distance between the top and base planes along the normal is

$$\gamma = -\frac{(\overline{V}-\overline{XB})\cdot\overline{N}}{\overline{H}\cdot\overline{N}}$$

and

$$0 \leq \gamma \leq 1$$

for an intersect to occur between the two planes of the TEC.

Let M equal the radius of the intersected ellipse along the major axis and m equal the radius of the intersected ellipse along the minor axis. Therefore, as shown in a partial cross-section of the TEC (Figure 21).

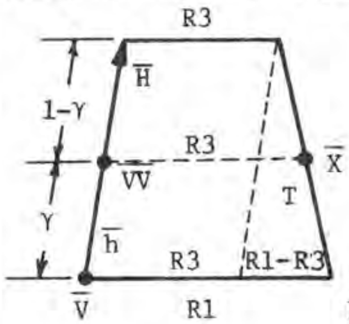


FIG. 21. TEC Cross-Section

$$\frac{1}{R1-R3} = \frac{1-\gamma}{T}$$

and

$$T = (R1-R3) - \gamma\,(R1-R3)$$

$$M = T + R3$$

Therefore

$$\boxed{M = \gamma R3 + R1\,(1-\gamma)} \tag{86}$$

It can similarly be shown that

$$\boxed{m = \gamma R4 + R2\,(1-\gamma)} \tag{87}$$

From the partial cross-section above it can also be seen that

$$\frac{1}{\overline{H}} = \frac{\gamma}{\overline{h}} \quad \text{or} \quad \overline{h} = \gamma\overline{H}$$

Therefore, the vertex, $\overline{VV}$, of the intersected ellipse is

$$\overline{VV} = \overline{V} + \gamma\overline{H} \tag{88}$$

and the vector from the intersect point to the vertex, $\overline{VV}$, of the intersect ellipse is

$$\bar{X} - \overline{VV} \tag{89}$$

Substituting ray Equation (85) and Equation (88) into Equation (89) results in

$$\overline{XB} + S\cdot\overline{WB} - \bar{V} - \gamma\bar{H} \tag{90}$$

The equation of the intersection ellipse is given by

$$\frac{(\bar{X}-\bar{h})^2}{M^2} + \frac{(\bar{Y}-\bar{k})^2}{m^2} = 1 \tag{91}$$

Since $\bar{A}$ is the semi-major axis unit vector of the base ellipse, the semi-minor axis unit vector of the base ellipse is

$$\bar{B} = \bar{A} \times \bar{N}$$

Therefore, from Equations (90) and (91)

$$\frac{\left[\left(\overline{XB} + S\cdot\overline{WB}-\bar{V}-\gamma\bar{H}\right)\cdot\bar{A}\right]^2}{M^2} + \frac{\left[\left(\overline{XB} + S\cdot\overline{WB}-\bar{V}-\gamma\bar{H}\right)\cdot\left(\bar{A}\times\bar{N}\right)\right]^2}{m^2} = 1$$

Solving for S and grouping terms for a quadratic in S results in

$$\boxed{\begin{aligned} &S^2\left[m^2\left(\overline{WB}\cdot\bar{A}\right)^2 + M^2\left(\overline{WB}\cdot\left(\bar{A}\times\bar{N}\right)\right)^2\right] \\ &-2S\left[m^2\left(\overline{WB}\cdot\bar{A}\right)\left(\bar{V}-\overline{XB} + \gamma\bar{H}\right)\cdot\bar{A} + M^2\cdot\overline{WB}\cdot\left(\bar{A}\times\bar{N}\right)\left(\left(\bar{V}-\overline{XB} + \gamma\bar{H}\right)\cdot\left(\bar{A}\times\bar{N}\right)\right)\right] \\ &+m^2\left(\left(\overline{XB}-\bar{V}-\gamma\bar{H}\right)\cdot\bar{A}\right)^2 + M^2\left(\left(\overline{XB}-\bar{V}-\gamma\bar{H}\right)\cdot\left(\bar{A}\times\bar{N}\right)\right)^2 - M^2m^2 = 0\end{aligned}} \tag{92}$$

Let

τ equal the coefficient of S^2

λ equal the coefficient of 2S

μ equal the constant term

Therefore Equation (92) becomes

$$\tau S^2 - 2\lambda S + \mu = 0$$

Solving for S

$$S = \frac{\lambda \pm \sqrt{\lambda^2 - \mu\tau}}{\tau}$$

If $(\lambda^2 - \mu\tau) < 0$, no intersection with the TEC occurs.

Case 1 specifies a ray parallel to the top and base planes. Therefore, from Equation (92), T cannot equal zero.

If $\sqrt{\lambda^2 - \mu\tau} > 0$ and $|\tau| > 0.0001$ then

$$\boxed{\text{RIN} = \frac{\lambda - \sqrt{\lambda^2 - \mu\tau}}{\tau}} \tag{93}$$

$$\boxed{\text{ROUT} = \frac{\lambda + \sqrt{\lambda^2 - \mu\tau}}{\tau}} \tag{94}$$

are the intersects with the quadratic surface if ROUT > RIN, $|\text{RIN-ROUT}| > 0.0001$, and ROUT > 0.

Case 2

Compute the intersect distances (if any), RIN and ROUT, for a ray that is not parallel to the planar surface of the TEC where

$$\overline{WB} \cdot \overline{N} \neq 0$$

Let α equal the distance along the ray from the plane of the base ellipse to the plane of the top ellipse. Therefore

$$\alpha = \frac{\overline{H}\cdot\overline{N}}{\overline{WB}\cdot\overline{N}}$$

Let β equal the distance along the ray from the start of the ray $(\overline{XB})$ to the plane of the base ellipse. Therefore

$$\beta = \frac{(\overline{V}-\overline{XB})\cdot\overline{N}}{\overline{WB}\cdot\overline{N}}$$

The distance along the ray from the start of the ray $(\overline{XB})$ to the plane of the top ellipse is given by

$$S_T = \alpha + \beta$$

Let θ equal the ratio of the distance between the base plane and the plane containing the intersect point to the distance between the base plane and the top plane (the plane containing the intersect point is parallel to the top and base planes). Therefore, the distance along the ray from the start of the ray $(\overline{XB})$ to the intersect plane or contact point is given by

$$\boxed{S = \alpha\theta + \beta} \tag{95}$$

From Equation (88) the vertex, $\overline{VV}$, of the intersected ellipse is

$$\overline{VV} = \overline{V} + \theta\overline{H} \tag{96}$$

and the vector from the intersect point to the vertex, $\overline{VV}$, of the intersect ellipse is

$$\overline{X} - \overline{VV} \tag{97}$$

Substituting ray Equation (85) and Equation (96) into Equation (97) results in

$$\overline{XB} + S\cdot\overline{WB}-\overline{V}-\theta\overline{H} \tag{98}$$

Since M is the radius of the intersected ellipse along the semi-major axis, and m is the radius of the intersected ellipse along the semi-minor axis

$$\frac{M}{m} = \frac{R1}{R2} = \frac{R3}{R4}$$

and

$$M^2 = m^2 \left(\frac{R1}{R2}\right)^2 \tag{99}$$

From Equation (87) it can be seen that

$$m = \theta R4 + R2\ (1-\theta)$$

or

$$m = R2 + \theta\ (R4-R2) \tag{100}$$

Substituting Equations (95), (98), (99), and (100) into Equation (91) results in

$$\frac{\left\{\left[\theta(\alpha\overline{WB}\cdot\overline{H})-(\overline{V}-\overline{XB}-\beta\overline{WB})\right]\cdot\overline{A}\right\}^2}{\left(\frac{R1}{R2}\right)^2\left[R2+\theta(R4-R2)\right]^2}+\frac{\left\{\left[\theta(\alpha\overline{WB}-\overline{H})-(\overline{V}-\overline{XB}-\beta\overline{WB})\right]\cdot\overline{B}\right\}^2}{\left[R2+\theta(R4-R2)\right]^2}=1$$

Solving for θ and grouping for a quadratic in θ results in

$$\boxed{\begin{aligned}&\theta^2\left[\left(\alpha\overline{WB}\cdot\overline{A}-\overline{H}\cdot\overline{A}\right)^2+\left({}^{R1}/_{R2}\right)^2\left(\alpha\overline{WB}\cdot\overline{B}-\overline{H}\cdot\overline{B}\right)^2-\left({}^{R1}/_{R2}\right)^2\left(R4-R2\right)^2\right]\\&-2\theta\left[\left(\alpha\overline{WB}\cdot\overline{A}-\overline{H}\cdot\overline{A}\right)\left(\left(\overline{V}-\overline{XB}\right)\cdot\overline{A}-\beta\overline{WB}\cdot\overline{A}\right)+\left({}^{R1}/_{R2}\right)^2\left(R2\right)\left(R4\right)-\left({}^{R1}/_{R2}\right)^2\left(R2\right)^2\right.\\&\left.+\left({}^{R1}/_{R2}\right)^2\left(\alpha\overline{WB}\cdot\overline{B}-\overline{H}\cdot\overline{B}\right)\left(\left(\overline{V}-\overline{XB}\right)\cdot\overline{B}-\beta\overline{WB}\cdot\overline{B}\right)\right]\\&+\left[\left(\left(\overline{V}-\overline{XB}\right)\cdot\overline{A}-\beta\overline{WB}\cdot\overline{A}\right)^2+\left({}^{R1}/_{R2}\right)^2\left(\left(\overline{V}-\overline{XB}\right)\cdot\overline{B}-\beta\overline{WB}\cdot\overline{B}\right)^2-\left({}^{R1}/_{R2}\right)^2\left(R2\right)^2\right]=0\end{aligned}} \tag{101}$$

Let

τ equal the coefficient of θ^2

λ equal the coefficient of 2θ

μ equal the constant term

Therefore Equation (101) becomes

$$\tau\theta^2+2\lambda\theta + \mu = 0 \tag{102}$$

Solving for θ

$$\theta = \frac{-\lambda \pm \sqrt{\lambda^2-\mu\tau}}{\tau} \tag{103}$$

If $\tau = 0$ the $\overline{H}$ vector is perpendicular to the base and the ray is parallel to the $\overline{H}$ vector, and the sides of the TEC are parallel as can be seen from Equation (101). Therefore, intersections can only occur with the planar surfaces.

If $(\lambda^2-\mu\tau) = 0$ the ray is parallel to a side of the TEC and there can be only one intersect with the quadratic surface. Therefore, from Equation (103)

$$\theta = \frac{\lambda}{\tau}$$

For two intersects with the quadratic surface

$$\boxed{\theta_1 = \frac{-\lambda + \sqrt{\lambda^2-\mu\tau}}{\tau}} \tag{104}$$

$$\boxed{\theta_2 = \frac{-\lambda - \sqrt{\lambda^2-\mu\tau}}{\tau}} \tag{105}$$

Choose such that $\theta_1 \geq \theta_2$

From Equation (95) candidates for RIN and ROUT are therefore

$$\boxed{\begin{array}{l} RIN = \alpha\theta_1 + \beta \\ ROUT = \alpha\theta_2 + \beta \end{array}} \qquad (106)$$

For a valid intersect the intersect S must occur between the base and top planes. Therefore

$$0 < S\left(\overline{WB}\cdot\overline{N}\right) - \left(\overline{V}-\overline{XB}\right)\cdot\overline{N} < \overline{H}\cdot\overline{N} \qquad (107)$$

must be true for a valid intersect with the quadratic surface.

When an intersect occurs with the $\overline{V}$ or base plane such that the vector $(\overline{X}-\overline{V})$ lies within the plane, the vector $(\overline{X}-\overline{V})$ is perpendicular to the normal $\overline{N}$. Therefore,

$$(\overline{X}-\overline{V})\cdot\overline{N} = 0$$

Substituting ray Equation (85)

$$\left(\overline{XB} + S_V\cdot\overline{WB}-\overline{V}\right)\cdot\overline{N} = 0$$

and

$$S_V = \frac{(\overline{V}-\overline{XB})\cdot\overline{N}}{\overline{WB}\cdot\overline{N}} = \beta \qquad (108)$$

Similarly for an intersect with the $\overline{V}+\overline{H}$ plane

$$\left[\overline{XB} + S_{V+H}\,\overline{WB} - (\overline{V}+\overline{H})\right]\cdot\overline{N} = 0$$

or

$$S_{V+H} = \frac{\overline{H}\cdot\overline{N}}{\overline{WB}\cdot\overline{N}} + \frac{(\overline{V}-\overline{XB})\cdot\overline{N}}{\overline{WB}\cdot\overline{N}} = \alpha + \beta \quad (109)$$

For a valid intersection with the base plane, the intersect must lie within the elliptic cross-section of the base. The intersect with respect to the semi-major axis $(\overline{A})$ is given by

$$F1 = S(\overline{WB}\cdot\overline{A}) - (\overline{V}-\overline{XB})\cdot\overline{A}$$

and the intersect with respect to the semi-minor axis $(\overline{B})$ is given by

$$F2 = S(\overline{WB}\cdot\overline{B}) - (\overline{V}-\overline{XB})\cdot\overline{B}$$

These values are substituted in the equation for an ellipse which must result in a value of one or less for a valid intersect as follows

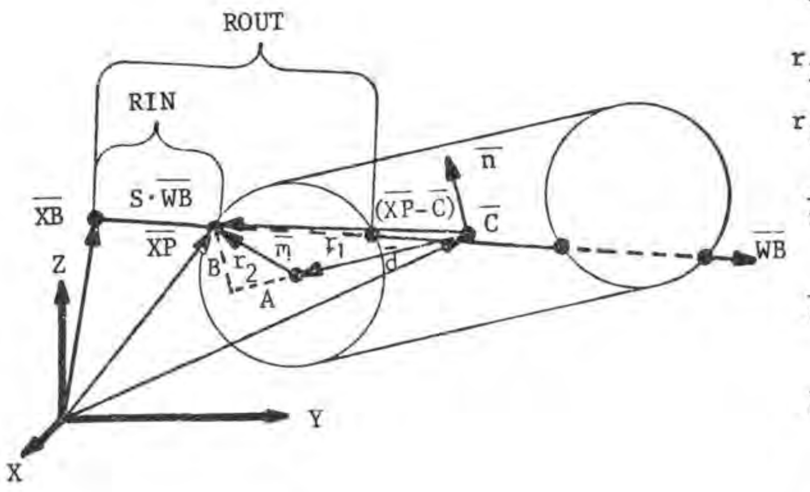
$$\frac{(F1)^2}{(R1)^2} + \frac{(F2)^2}{(R2)^2} \leq 1 \quad (110)$$

An intersect with the top ellipse is tested in the same manner except radii R3 and R4 are used in Equation (110).

TORUS (SUBROUTINE TOR)

Subroutine TOR calculates the intersection distances (if any) of a ray in space and a torus, as shown in Figure 21. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equation defining the torus.

The torus is defined by a center, $\overline{C}$; a radius, r_1, from the center of the torus to the midpoint of the solid circular portion; a radius, r_2, of the circular cross-section; and a normal, $\overline{n}$, to the plane of the torus. Let $\overline{d}$ be a unit vector in the direction of r_1 at the intersect point, as shown in Figure 22.



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{XP}$	point of intersect with torus
$\overline{C}$	center of torus
r_1	radius of torus
r_2	radius of circular cross-section
$\overline{n}$	normal to plane of torus (unit vector)
$\overline{d}$	direction of r_1 (unit vector)
$\overline{m}$	vector from midpoint of cross-section to point of contact

$$\overline{n}\cdot\overline{n} = 1$$

$$\overline{d}\cdot\overline{d} = 1$$

$$\overline{m}\cdot\overline{m} = r_2^2$$

$$\overline{d}\cdot\overline{n} = 0$$

$$\overline{XP} = \overline{XB}+S\cdot\overline{WB}$$

FIG. 22. Torus

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB} \cdot S \tag{111}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (111) will give a point $\overline{X}$ (x, y, z) along the ray, as shown in the figure.

From Figure 22, it can be seen that

$$\overline{m} = (\overline{XP}-\overline{C}) - r_1\overline{d}$$

Since

$$\overline{m} \cdot \overline{m} = r_2^2$$

and

$$\overline{XP} = \overline{XB} + \overline{WB} \cdot S$$

then

$$[(\overline{XB}+S\cdot\overline{WB}-\overline{C}) - r_1\overline{d}] \cdot [(\overline{XB}+S\cdot\overline{WB}-\overline{C}) - r_1\overline{d}] = r_2^2$$

Expanding and rearranging results in:

$$(\overline{XB}+S\cdot\overline{WB}-\overline{C})\cdot\overline{d} = \frac{r_2^2 - r_1^2 - (\overline{XB}+S\cdot\overline{WB}-\overline{C})^2}{-2\ r_1} \tag{112}$$

Also from Figure 22 it can be seen that

$$A^2 + B^2 = r_2^2 \tag{113}$$

and

$$A = (\overline{XB}+S\cdot\overline{WB}-\overline{C})\cdot\overline{d} - r_1$$

$$B = (\overline{XB}+S\cdot\overline{WB}-\overline{C})\cdot\overline{n}$$

Substituting A and B into Equation (113) results in

$$[(\overline{XB}+S\cdot\overline{WB}-\overline{C})\cdot\overline{n}]^2 + [(\overline{XB}+S\cdot\overline{WB}-\overline{C})\cdot\overline{d}-r_1]^2 = r_2^2 \tag{114}$$

Substituting Equation (112) into Equation (114) and rearranging terms gives

$$4r_1^2[S\cdot\overline{WB}\cdot\overline{n}+(\overline{XB}-\overline{C})\cdot\overline{n}]^2 + [(\overline{XB}+S\cdot\overline{WB}-\overline{C})^2 - (r_1^2+r_2^2)]^2 = 4r_1^2r_2^2 \tag{115}$$

By expanding Equation (115), realizing that $\overline{WB}\cdot\overline{WB}=1$, and regrouping terms for a quartic equation in 'S' of the form

$$AS^4 + BS^3 + CS^2 + DS + E = 0 \tag{116}$$

results in the following coefficients for Equation (116)

$$\boxed{\begin{aligned}
A &= 1\\
B &= 4\,\overline{WB}\cdot(\overline{XB}-\overline{C})\\
C &= 4r_1^2(\overline{WB}\cdot\overline{n})^2 + 4[\overline{WB}\cdot(\overline{XB}-\overline{C})]^2 + 2[(\overline{XB}-\overline{C})^2 - (r_1^2 + r_2^2)]\\
D &= 8r_1^2(\overline{WB}\cdot\overline{n})[(\overline{XB}-\overline{C})\cdot\overline{n}] + 4[\overline{WB}\cdot(\overline{XB}-\overline{C})]\,[(\overline{XB}-\overline{C})^2 - (r_1^2 + r_2^2)]\\
E &= 4r_1^2\,[(\overline{XB}-\overline{C})\cdot\overline{n}]^2 + [(\overline{XB}-\overline{C})^2 - (r_1^2 + r_2^2)]^2 - 4\,r_1^2\,r_2^2
\end{aligned}} \tag{117}$$

With the values of A, B, C, D, and E known, Subroutine TOR calls Subroutine QRTIC to find the roots and number of real roots of Equation (116).

Four real roots means that there are two sets of entry and exit intersects, or two sets of RIN and ROUT.

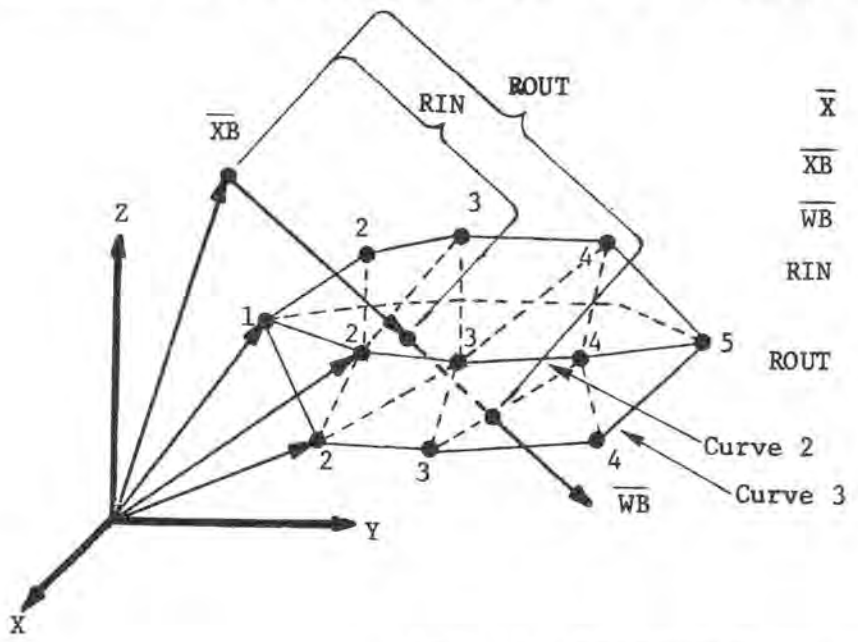
Two real roots means that there is one set of entry and exit intersects, or one set of RIN and ROUT.

No real roots means that the ray does not intersect the torus.

ARBITRARY SURFACE (SUBROUTINE ARS)

Subroutine ARS calculates the intersection distances (if any) of a ray in space and an arbitrary surface. The intersection distances, RIN and ROUT, are calculated by dividing the arbitrary surface into triangles and solving simultaneously the ray equation and the equation of each triangle on the arbitrary surface that lies within a given tolerance of the ray.

The arbitrary surface is described by a specified number of curves and a specified number of points on each curve. A surface is constructed between the first curve and the second curve, between the second curve and the third curve, etc. Subroutine ARS computes triangles on the surface formed between two curves by using the points on the curves as shown in Figure 23.



$\overline{X}$	intersect point
$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 23. Arbitrary Surface

Any intersect $\overline{X}$ along ray $\overline{WB}$ from point $\overline{XB}$ is given by

$$\overline{X} = \overline{XB} + \overline{WB}\cdot S \tag{118}$$

where

$\overline{XB}$ (x_o, y_o, z_o) is a fixed point on the ray.

$\overline{WB}$ (W_x, W_y, W_z) is the direction cosines of the ray.

Thus, for any value of the scalar, S, Equation (118) will give a point $\overline{X}$ (x,y,z) along the ray.

Subroutine ARS divides the surface between two adjacent curves into triangles using the specified points on each curve. The intersect distance and the normal (if any) are computed for each triangle as shown in the following figure of a single triangle from the arbitrary surface.

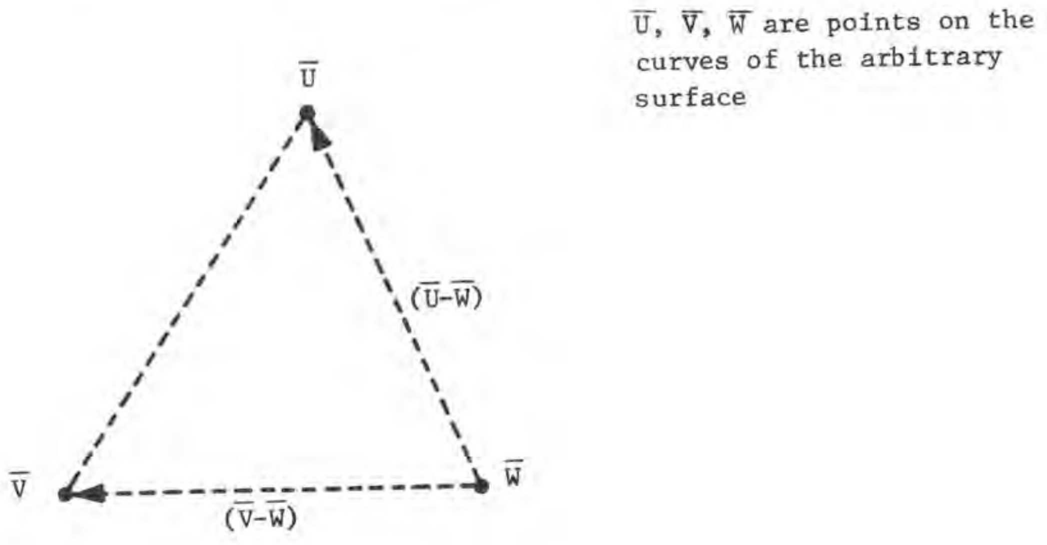

$\overline{U}$, $\overline{V}$, $\overline{W}$ are points on the curves of the arbitrary surface
$\overline{U}$
$(\overline{U}-\overline{W})$
$\overline{V}$
$\overline{W}$
$(\overline{V}-\overline{W})$

FIG. 24. ARS Surface Triangle

For the intersect $\overline{XP}$ within the triangle there is an α, β, and γ such that

$$\boxed{\overline{XP} = \overline{XB} + \overline{WB}\cdot S = \alpha\overline{U} + \beta\overline{V} + \gamma\overline{W}} \tag{119}$$

where

$$\alpha + \beta + \gamma = 1$$

and

$$0 \leq \alpha \leq 1$$

$$0 \leq \beta \leq 1$$

$$0 \leq \gamma \leq 1$$

Substituting $\gamma = 1-\alpha-\beta$ into Equation (119) results in

$$\alpha\overline{U} + \beta\overline{V} + (1-\alpha-\beta)\overline{W} = \overline{XB} + S\cdot\overline{WB}$$

Collecting terms and rearranging results in

$$\alpha(\overline{U}-\overline{W}) + \beta(\overline{V}-\overline{W}) - S\cdot\overline{WB} = \overline{XB}-\overline{W} \tag{120}$$

Equation(120)can be written in terms of the x, y, and z components as a matrix equation with three unknowns as follows.

$$\boxed{\begin{aligned} \alpha(\overline{U}_x-\overline{W}_x) + \beta(\overline{V}_x-\overline{W}_x) - S\cdot\overline{WB}_x &= \overline{XB}_x-\overline{W}_x \\ \alpha(\overline{U}_y-\overline{W}_y) + \beta(\overline{V}_y-\overline{W}_y) - S\cdot\overline{WB}_y &= \overline{XB}_y-\overline{W}_y \\ \alpha(\overline{U}_z-\overline{W}_z) + \beta(\overline{V}_z-\overline{W}_z) - S\cdot\overline{WB}_z &= \overline{XB}_z-\overline{W}_z \end{aligned}} \tag{121}$$

Solving Equation(121)results in values for α, β,γ ($\gamma = 1-\alpha-\beta$), and S. If α, β, and γ satisfy the conditions $\alpha + \beta + \gamma = 1$ and $0 \leq (\alpha,\beta,\gamma) \leq 1$, the ray intersects the triangle.

The normal of the intersected triangle is computed from the cross product of two sides of the triangle, or

$$\overline{N} = (\overline{U}-\overline{W}) \times (\overline{V}-\overline{W})$$

Subroutine ARS computes the normal of the intersected triangle such that its direction is outward from the arbitrary surface. The direction of the resultant normal is compared with the direction of the ray to determine if the intersected triangle is an entry or exit intersect.

ANGLE OF OBLIQUITY (SUBROUTINE CALC)

Given the body type and the intersected surface number (positive for an entry intersect, negative for an exit intersect) where a ray with direction cosines $\overline{WS}$ intersects the body, Subroutine CALC calculates the angle of obliquity, that is, the angle between the ray and the normal at the intersect point.

To compute the obliquity angle, the direction cosines of the normal $\overline{WB}$, must be computed so that $\overline{WB} \cdot \overline{WS} > 0$. Variable Q (XNOS in simulation model) is set +1 or -1 to insure a $\overline{WB}$ such that $\overline{WB} \cdot \overline{WS} > 0$.

The methods for computing the normal direction cosines, $\overline{WB}$, for each of the twelve body types are given in the following paragraphs.

Normal to the Rectangular Parallelepiped

The direction cosines, $\overline{WB}$, of the normal to the intersected surface of an RPP are determined from the fact that the bounding planes are parallel to the coordinate axes as shown in Figure 25.

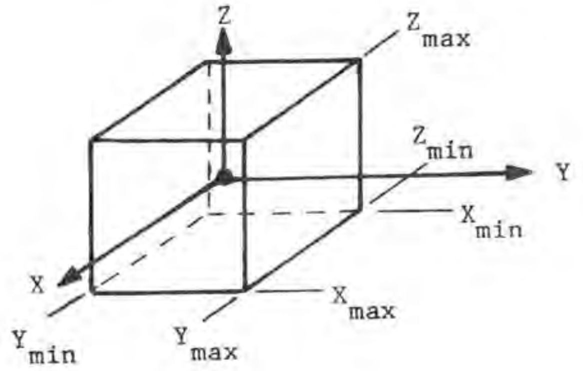


FIG. 25. Rectangular Parallelepiped

Surface Number	Bounding Plane	Direction Cosines of Normal into RPP [$\overline{WB}$ (x,y,z)]
1	X min	(1, 0, 0)
2	X max.	(-1, 0, 0)
3	Y min	(0, 1, 0)
4	Y max.	(0, -1, 0)
5	Z min	(0, 0, 1)
6	Z max.	(0, 0, -1)

Normal to the Box (Or Sides 1,3,5, or 6 of the Right Angle Wedge)

The direction cosines of the normal to the intersected surface of a BOX is determined from the vector between the face pairs, one of which is the intersected face, as shown in Figure 26.

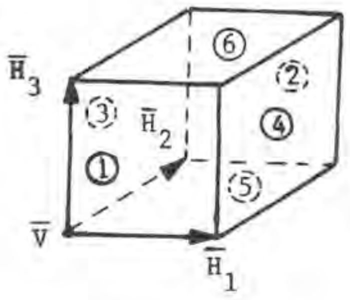


FIG. 26. Box

Surface Number	Direction Cosines	
1 or 2	$\overline{WB}_x = Q\ (\overline{H2}/\sqrt{\overline{H2}\cdot\overline{H2}})$	(122)
3 or 4	$\overline{WB}_y = Q\ (\overline{H1}/\sqrt{\overline{H1}\cdot\overline{H1}})$	(123)
5 or 6	$\overline{WB}_z = Q\ (\overline{H3}/\sqrt{\overline{H3}\cdot\overline{H3}})$	(124)

Where $\sqrt{\overline{H}\cdot\overline{H}}$ is the scalar length of the given vector, and Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect. If the box surface number is even, or if surface number six of the RAW, the direction of the normal at the intersect is opposite the direction of the vector between face pairs so Q is set to -Q.

Normal to the Sphere

The direction cosines of the normal to the intersected surface of a sphere are determined from intersect point $\overline{X}$, vertex $\overline{V}$, and radius R, as shown in Figure 27.

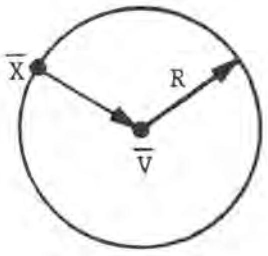


FIG. 27 Sphere

$$\overline{WB}_x = \frac{\overline{V}_x - \overline{X}_x}{R} \; Q \tag{125}$$

$$\overline{WB}_y = \frac{\overline{V}_y - \overline{X}_y}{R} \; Q \tag{126}$$

$$\overline{WB}_z = \frac{\overline{V}_z - \overline{X}_z}{R} \; Q \tag{127}$$

where Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect.

Normal to a Planar Surface of the Right Circular Cylinder, The Right Elliptical Cylinder, or the Truncated Right Cone

The direction cosines of the normal to the intersected planar surface of an RCC or REC are determined from height vector $\overline{H}$ as follows

$$\overline{WB}_x = \frac{\overline{H}_x}{\sqrt{\overline{H}\cdot\overline{H}}} \; Q \tag{128}$$

$$\overline{WB}_y = \frac{\overline{H}_y}{\sqrt{\overline{H}\cdot\overline{H}}} \; Q \tag{129}$$

$$\overline{WB}_z = \frac{\overline{H}_z}{\sqrt{\overline{H}\cdot\overline{H}}} \; Q \tag{130}$$

where $\sqrt{\overline{H}\cdot\overline{H}}$ is the scalar length of the height vector and Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect. If the $\overline{V}+\overline{H}$ planar surface is the intersected surface, Q is set to -Q before computing the direction cosines of the normal since the direction of the normal to the $\overline{V}+\overline{H}$ planar surface is opposite the direction of the $\overline{H}$ vector.

Normal to the Side of the Right Circular Cylinder

The direction cosines of the normal for an intersect on the side of an RCC are determined from intersect point $\overline{X}$, vertex $\overline{V}$, and height vector $\overline{H}$, as shown in Figure 28

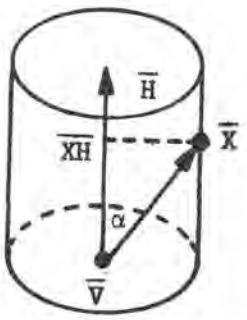


FIG. 28. Right Circular Cylinder

The direction cosines of a vector from $\overline{V}$ to $\overline{X}$ are given by

$$\overline{WN} = \frac{\overline{X}-\overline{V}}{\sqrt{\Sigma(\overline{V}_i-\overline{X}_i)^2}} \tag{131}$$

where i is the x, y, or z coordinate respectively.

The direction cosines of height vector $\overline{V}$ are given by

$$\overline{WI} = \frac{\overline{H}}{\sqrt{\overline{H}\cdot\overline{H}}} \tag{132}$$

The cosine of an angle α between the two unit vectors $\overline{WN}$ and $\overline{WH}$ is given by

$$\cos\alpha = \overline{WN}\cdot\overline{WI}$$

Therefore, the distance from vertex $\overline{V}$ to the intersect projected onto the height vector $\overline{H}$ is

$$D = \sqrt{(\overline{X}-\overline{V})\cdot(\overline{X}-\overline{V})}\ \cos\alpha \tag{133}$$

and the coordinates of the intersect projected onto the height vector $\overline{H}$ are given by

$$\overline{XH} = \overline{V} + D\cdot\overline{WI} \tag{134}$$

The direction cosines of the normal to the side of the RCC are therefore determined from $\overline{X}$ and $\overline{XH}$ as follows

$$\overline{WB}_x = \frac{\overline{XH}_x - \overline{X}_x}{\sqrt{\Sigma(\overline{X}_i - \overline{XH}_i)^2}} \quad Q \tag{135}$$

$$\overline{WB}_y = \frac{\overline{XH}_y - \overline{X}_y}{\sqrt{\Sigma(\overline{X}_i - \overline{XH}_i)^2}} \quad Q \tag{136}$$

$$\overline{WB}_z = \frac{\overline{XH}_z - \overline{X}_z}{\sqrt{\Sigma(\overline{X}_i - \overline{XH}_i)^2}} \quad Q \tag{137}$$

where i is the x, y, or z coordinate, respectively, and Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect.

Normal to the Side of the Right Elliptic Cylinder

The direction cosines of the normal for an intersect on the side of an REC are determined from intersect $\overline{X}$, the length and direction of semi-major axis $\overline{A}$, the length and direction of semi-minor axis $\overline{B}$, and the intersect projected onto height vector $\overline{XH}$ ($\overline{XH}$ was determined in the preceding RCC section) as shown in Figure 29.

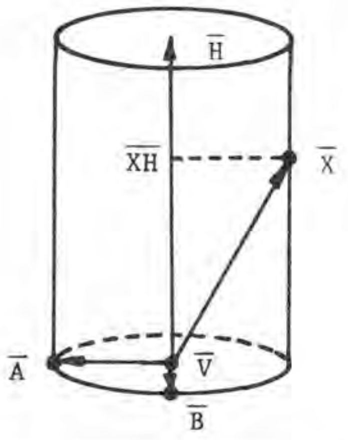


FIG. 29. Right Elliptic Cylinder

The distance, a, between $\overline{XH}$ and $\overline{A}$ is given by

$$a = \sqrt{(\overline{XH} - \overline{A})^2} \tag{138}$$

The distance, b, between $\overline{XH}$ and $\overline{B}$ is given by

$$b = \sqrt{(\overline{XH}-\overline{B})^2} \tag{139}$$

The distance, c, from the center, $\overline{XH}$, of the intersection ellipse to a focus $\overline{F1}$ is given by

$$c = \sqrt{a^2-b^2} \tag{140}$$

where a > b.

The direction cosines, $\overline{WA}$, of the semi-major axis are given by

$$\overline{WA} = \frac{\overline{A}_i-\overline{XH}_i}{\sqrt{\Sigma(\overline{XH}_i-\overline{A}_i)^2}} \tag{141}$$

where i is the x, y, or z coordinate, respectively.

The coordinates of the foci, $\overline{F1}$ and $\overline{F2}$, are determined by

$$\overline{F1} = \overline{XH} + C\cdot\overline{WA}$$

$$\overline{F2} = \overline{XH} - C\cdot\overline{WA}$$

The directions cosines, $\overline{WX}$, from focus $\overline{F1}$ to the intersect $\overline{X}$ are given by

$$\overline{WX} = \frac{\overline{X}_i-\overline{F1}_i}{\sqrt{\Sigma(\overline{F1}_i-\overline{X}_i)^2}} \tag{142}$$

where i is the x, y, or z coordinate, respectively.

A point, $\overline{L}$, on a line from $\overline{F1}$ through intersect $\overline{X}$ at a distance of twice the length of the semi-major axis is given by

$$\overline{L} = \overline{F1} + 2a\cdot\overline{WX}$$

The direction cosines of a vector from point $\overline{L}$ to the second focus $\overline{F2}$ are equal to the direction cosines, $\overline{WB}$, of the normal to the surface at intersect $\overline{X}$.

$$\overline{WB}_x = \frac{\overline{F2}_x-\overline{L}_x}{\sqrt{\Sigma(\overline{L}_i-\overline{F2}_i)^2}} \quad Q \tag{143}$$

$$\overline{WB}_y = \frac{\overline{F2}_y - \overline{L}_y}{\sqrt{\Sigma(\overline{L}_i - \overline{F2}_i)^2}} \quad Q \tag{144}$$

$$\overline{WB}_3 = \frac{\overline{F2}_z - \overline{L}_z}{\sqrt{\Sigma(\overline{L}_i - \overline{F2}_i)^2}} \quad Q \tag{145}$$

where i is the x, y, or z coordinate, respectively, and Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect.

Normal to the Side of the Truncated Right Cone

The direction cosines of the normal for an intersected surface of the side of a TRC are determined from intersect point $\overline{X}$, vertex $\overline{V}$, height vector $\overline{H}$, radius RB of the base, and radius RT of the top as shown in Figure 30.

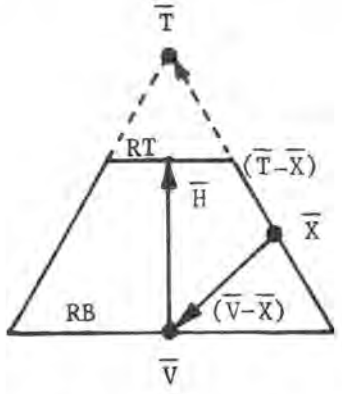


FIG. 30. Truncated Right Cone

The coordinates of point $\overline{T}$ are determined by

$$\overline{T} = \overline{V} + \frac{RB}{RB-RT}\,\overline{H} \tag{146}$$

The cross product of vectors $(\overline{T}-\overline{X})$ and $(\overline{V}-\overline{X})$ results in a vector $\overline{CP}$ by

$$\overline{CP} = (\overline{T}-\overline{X}) \times (\overline{V}-\overline{X}) \tag{147}$$

Therefore, a vector perpendicular to the side of the intersect is

$$\overline{N} = \overline{CP} \times (\overline{T}-\overline{X}) \tag{148}$$

and the direction cosines of the normal at the intersect are determined by

$$\overline{WB}_x = \frac{\overline{N}_x}{\sqrt{\overline{N}\cdot\overline{N}}} Q \tag{149}$$

$$\overline{WB}_y = \frac{\overline{N}_y}{\sqrt{\overline{N}\cdot\overline{N}}} Q \tag{150}$$

$$\overline{WB}_z = \frac{\overline{N}_z}{\sqrt{\overline{N}\cdot\overline{N}}} Q \tag{151}$$

where $\sqrt{\overline{N}\cdot\overline{N}}$ is the scalar length of the perpendicular vector at the intersect, and Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect.

Normal to the Ellipsoid of Revolution

The directions cosines of the normal for an intersect on the surface of an ellipsoid are determined from intersect point $\overline{X}$, foci $\overline{F1}$ and $\overline{F2}$, and the length of semi-major axis A, as shown in Figure 31.

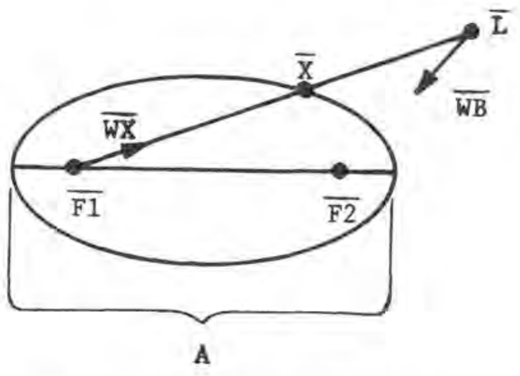


FIG. 31. Ellipsoid of Revolution

The direction cosines, $\overline{WX}$ from focus $\overline{F1}$ to the intersect $\overline{X}$ are given by

$$\overline{WX} = \frac{\overline{X}_i - \overline{F1}_i}{\sqrt{\Sigma(\overline{F1}_i - \overline{X}_i)^2}} \tag{152}$$

A point, $\overline{L}$, on a line from $\overline{F1}$ through intersect $\overline{X}$ at a distance equal to the length of semi-major axis, A, is given by

$$\overline{L} = \overline{F1} + A \cdot \overline{WX}$$

The direction cosines of a vector from point $\overline{L}$ to the second focus $\overline{F2}$ are equal to the direction cosines, $\overline{WB}$, of the normal at intersect $\overline{X}$ as follows

$$\overline{WB}_x = \frac{\overline{F2}_x - \overline{L}_x}{\sqrt{\Sigma(\overline{L}_i - \overline{F2}_i)^2}} \quad Q \tag{153}$$

$$\overline{WB}_y = \frac{\overline{F2}_y - \overline{L}_y}{\sqrt{\Sigma(\overline{L}_i - \overline{F2}_i)^2}} \quad Q \tag{154}$$

$$\overline{WB}_z = \frac{\overline{F2}_z - \overline{L}_z}{\sqrt{\Sigma(\overline{L}_i - \overline{F2}_i)^2}} \quad Q \tag{155}$$

where i is the x, y, or z coordinate, respectively, and Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect.

Normal to the Slanted Surface of the Right Angle Wedge

The direction cosines of the normal for an intersect on the slanted surface of a RAW are determined from length vectors $\overline{H1}$, $\overline{H2}$, and $\overline{H3}$ as shown in Figure 32.

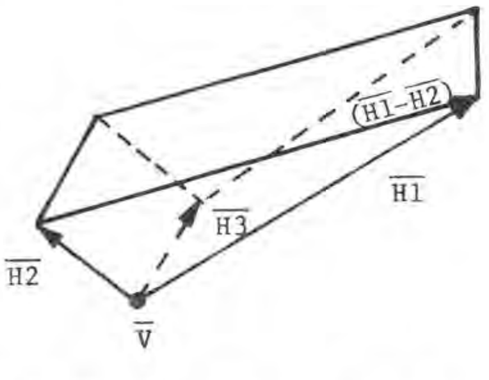


FIG. 32. Right Angle Wedge

A vector, $\overline{N}$, perpendicular to the slanted side is given by

$$\overline{N} = (\overline{H1}-\overline{H2})\times\overline{H3} \tag{156}$$

The length of $\overline{N}$ is given by

$$n = \sqrt{\overline{N}\cdot\overline{N}}$$

To insure that vector $\overline{N}$ is directed from the outside into the slanted surface (since vectors $\overline{H1}$ and $\overline{H2}$ may be interchanged), the dot product of $\overline{N}$ and $\overline{H1}$ is determined

$$S' = \overline{N}\cdot\overline{H1}$$

where

$S'<0$ means that $\overline{N}$ is in the correct direction

and

$S'>0$ means that $\overline{N}$ is in the reverse direction

The direction cosines of the normal to the slanted surface are therefore

$$\overline{WB} = \frac{\overline{N}}{n}\,(Q)(S) \tag{157}$$

where $S = -S'/|S'|$, and Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect.

Normal to the Arbitrary Polyhedron

The direction cosines of the normal for an intersect on the side of a ARB are determined from A, B, and C, the respective X, Y, and Z coefficients of the equation of the intersected plane, as shown in Figure 33.

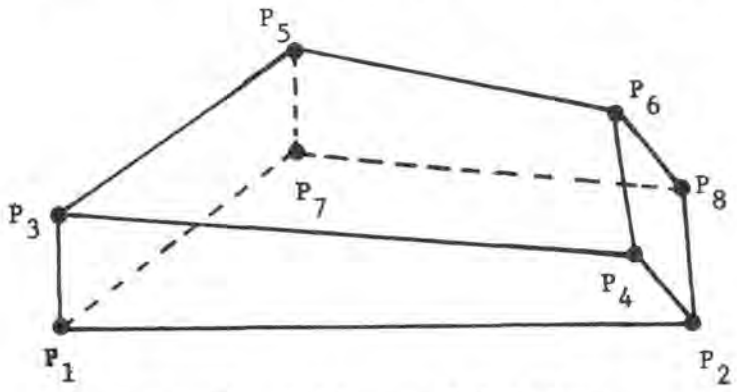


FIG. 33. Arbitrary Polyhedron

The direction cosines of the normal to the plane defined by AX + BY + CZ + D = 0 are given by

$$\overline{WB}_x = \frac{A}{\sqrt{A^2+B^2+C^2}} \quad Q \tag{158}$$

$$\overline{WB}_y = \frac{B}{\sqrt{A^2+B^2+C^2}} \quad Q \tag{159}$$

$$\overline{WB}_z = \frac{C}{\sqrt{A^2+B^2+C^2}} \quad Q \tag{160}$$

where Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect.

<u>Normal to the Truncated Elliptic Cone</u>

The direction cosines of the normal to either intersected planar surface of a TEC are determined from $\overline{N}$, the normal direction cosines to the base ellipse, as follows:

$$\overline{WB}_x = \overline{N}_x \cdot (Q) \tag{161}$$

$$\overline{WB}_y = \overline{N}_y \cdot (Q) \tag{162}$$

$$\overline{WB}_z = \overline{N}_z \cdot (Q) \tag{163}$$

where Q is set to a +1.0 for an entry intersect or -1.0 for an exit intersect. If the top planar surface is the intersected surface, Q is set to -Q before computing the direction cosines of the normal since the direction of the normal to the $\overline{V+H}$ planar surface is opposite the direction of the $\overline{N}$ vector.

The direction cosines of the normal for an intersect on the side of a TEC are determined from intersect $\overline{X}$, vertex $\overline{V}$, height vector $\overline{H}$, normal to the base ellipse $\overline{N}$, the direction cosines of semi-major $\overline{A}$, the direction cosines of semi-minor axis $\overline{B}$, the semi-major and semi-minor radii of base ellipse R1 and R2, and the semi-minor and semi-minor radii of top ellipse R3 and R4, as shown in Figure 34.

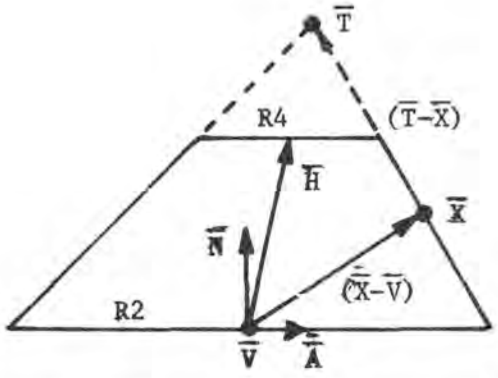


FIG. 34 Truncated Elliptic Cone

The distance between the two planar surfaces is given by

$$D1 = \overline{H}\cdot\overline{N}$$

The height of the intersect $\overline{X}$ from the base plane is given by

$$D2 = (\overline{X}-\overline{V})\cdot\overline{N}$$

The ratio on the normal to the height of the hit is given by

$$\gamma = D2/D1$$

Therefore, the center of the intersection ellipse is determined by

$$\overline{C} = \overline{V} + \gamma\cdot\overline{H}$$

By redrawing part of Figure 34 as shown in Figure 35:

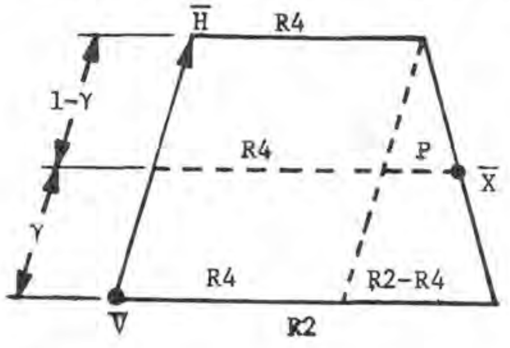


FIG. 35. TEC Cross-Section

it can be seen that

$$\frac{1-\gamma}{P} = \frac{1}{R2-R4}$$

or

$$P = R2-R4-\gamma R2+\gamma R4$$

If b is equal to one-half the length of the semi-minor axis of the intersected ellipse

$$b = R4+P$$

Therefore

$$b^2 = [\gamma R4 + (1-\gamma)\ R2]^2$$

Letting τ equal

$$\tau = \left(\frac{R1}{R2}\right)^2$$

Therefore, if a is equal to one-half the length of the semi-major axis of the intersected ellipse

$$a^2 = \tau b^2$$

The distance from the center of the intersection ellipse to a focus is given by

$$c = \sqrt{a^2-b^2}$$

The coordinates of foci $\overline{F1}$ and $\overline{F2}$ are

$$\overline{F1} = \overline{C} + c\overline{A}$$

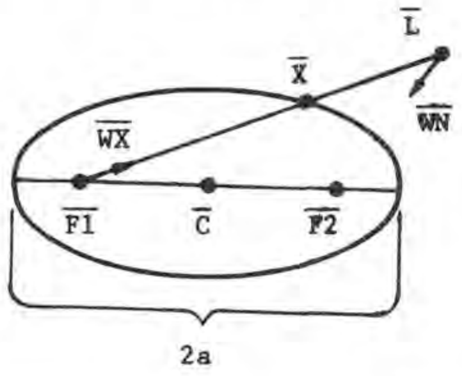
$$F2 = \overline{C} - c\overline{A}$$

The direction cosines, $\overline{WX}$, from focus $\overline{F1}$ to the intersect $\overline{X}$ are given by

$$\overline{WX} = \frac{\overline{X}_i-\overline{F1}_i}{\sqrt{\Sigma(\overline{F1}_i-\overline{X}_i)^2}} \qquad (164)$$

where i is equal to the x, y, or z coordinate.

From Figure 36, the point $\overline{L}$ on a line from $\overline{F1}$ through intersect $\overline{X}$ at a distance equal to the length of semi-major axis 2a is given by



$$\overline{L} = \overline{F1} + 2a\ \overline{WX}$$

FIG. 36. TEC Intersection Ellipse

The direction cosines of a vector from point $\overline{L}$ to the second focus $\overline{F2}$ are equal to the direction cosines, $\overline{WN}$, of the normal to the plane of the ellipse at intersect $\overline{X}$ as follows:

$$\overline{WN} \doteq \frac{\overline{F2}_i - \overline{L}_i}{\sqrt{\Sigma(\overline{L}_i - \overline{F2}_i)^2}} \tag{165}$$

where i is the x, y, or z coordinate, respectively.

From Figure 36, the vector $(\overline{T}-\overline{X})$ is determined by

$$(\overline{T}-\overline{X}) = \left(\overline{V} + \frac{R2}{R2-R4}\ \overline{H}\right) - \overline{X}$$

The cross product of the vectors $(\overline{T}-\overline{X})$ and $\overline{WN}$ results in a vector $\overline{CP}$ by

$$\overline{CP} = (\overline{T}-\overline{X}) \times \overline{WN} \tag{166}$$

Therefore, a vector perpendicular to the side of intersect $\overline{X}$ is determined by

$$\overline{N} = \overline{CP} \times (\overline{T}-\overline{X}) \tag{167}$$

and the direction cosines of the normal at the intersect are determined by

$$\overline{WB}_x = \frac{\overline{N}_x}{\sqrt{\overline{N}\cdot\overline{N}}}\ Q \tag{168}$$

$$\overline{WB}_y = \frac{\overline{N}_y}{\sqrt{\overline{N}\cdot\overline{N}}}\ Q \tag{169}$$

$$\overline{WB}_z = \frac{\overline{N}_z}{\sqrt{\overline{N}\cdot\overline{N}}}\ Q \tag{170}$$

where $\sqrt{\overline{N}\cdot\overline{N}}$ is the scalar length of the perpendicular vector at the intersect, and Q is set to a +1.0 for an entry intersect or a -1.0 for an exit intersect.

Normal to the Torus

The direction cosines of the normal to the intersected surface of a torus are determined from intersect $\overline{X}$, normal to the plane of the torus $\overline{n}$ center of the torus $\overline{V}$, and major radius R, as shown in Figure 37.

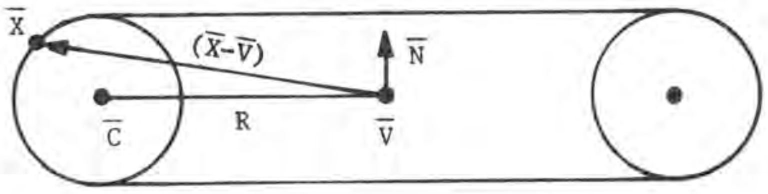


FIG. 37. Torus

The cross product of vectors $\overline{N}$ and $(\overline{X}-\overline{V})$ is given by

$$\overline{VX} = \overline{N} \times (\overline{X}-\overline{V})$$

The cross product $\overline{VX}$ and $\overline{N}$ is given by

$$\overline{CV} = \overline{VX} \times \overline{N}$$

and results in a vector in the direction of the center of the circular cross-section of the intersect plane.

The coordinates of point $\overline{C}$ are determined by

$$\overline{C} = \overline{V} + R \frac{\overline{CV}}{\sqrt{\overline{CV} \cdot \overline{CV}}}$$

where $\overline{CV}/\sqrt{\overline{CV} \cdot \overline{CV}}$ are the direction cosines of the vector $\overline{CV}$.

Knowing the coordinates of point $\overline{C}$ and intersect $\overline{X}$, the normal to the surface of intersect $\overline{X}$ is determined by

$$\overline{WB}_x = \frac{\overline{C}_x - \overline{X}_x}{\sqrt{\Sigma(\overline{X}_i - \overline{C}_i)^2}} Q \tag{171}$$

$$\overline{WB}_y = \frac{\overline{C}_y - \overline{X}_y}{\sqrt{\Sigma(\overline{X}_i - \overline{C}_i)^2}} Q \tag{172}$$

$$\overline{WB}_z = \frac{\overline{C}_z - \overline{X}_z}{\sqrt{\Sigma(\overline{X}_i - \overline{C}_i)^2}} Q \tag{173}$$

where i is the x, y, or z coordinate, respectively, and Q is set to +1.0 for an entry intersect or -1.0 for an exit intersect.

Normal to the Arbitrary Surface

The direction cosines of a normal to the intersected surface of an ARS are determined from the three points, $\overline{U}$, $\overline{V}$, and $\overline{W}$, of the intersected triangle on the surface of the ARS, as shown in Figure 38.

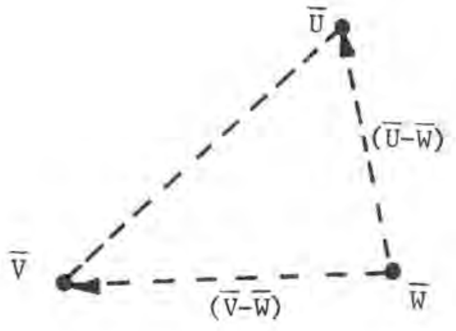


FIG. 38. ARS Surface Triangle

A vector $\overline{N}$ perpendicular to the surface of a triangle formed by the three points $\overline{U}$, $\overline{V}$, and $\overline{W}$ is given by

$$\overline{N} = (\overline{U}-\overline{W}) \times (\overline{V}-\overline{W})$$

The directions cosines of the normal vector $\overline{N}$ are determined by

$$\overline{WB}_x = \frac{\overline{N}_x}{\sqrt{\overline{N}\cdot\overline{N}}} \tag{174}$$

$$\overline{WB}_y = \frac{\overline{N}_y}{\sqrt{\overline{N}\cdot}} \tag{175}$$

$$\overline{WB}_z = \frac{\overline{N}_z}{\sqrt{\overline{N}\cdot\overline{N}}} \tag{176}$$

where $\sqrt{\overline{N}\cdot\overline{N}}$ is the scalar length of the normal vector $\overline{N}$.

Angle of Obliquity

The angle of obliquity (the angle between the ray and the normal at the intersect point) is determined from the direction cosines of the ray, $\overline{WS}$, and the direction cosines of the normal, $\overline{WB}$, as follows:

The cosine of the angle between the normal $\overline{WB}$ and the ray $\overline{WS}$ is

$$A = \overline{WB}\cdot\overline{WS} = |\overline{WB}|\cdot|\overline{WS}| \cos\alpha$$

Since $|\overline{WB}|$ and $|\overline{WS}|$ are both equal to one

$$A = \cos \alpha$$

The angle in degrees is determined by

$$\alpha(\text{degrees}) = \left[\arctan\left(\frac{\sqrt{1-(\cos \alpha)^2}}{\cos \alpha}\right)\right] \frac{180}{\pi} \tag{177}$$

where

$$\alpha\ (\text{degrees}) = \arctan\left(\frac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}\right)\frac{180}{\pi} = \arccos\ (\alpha)\ \frac{180}{\pi}$$

since the arctangent is the only standard FORTRAN inverse trigonometric routine.

GRID PLANE AND TARGET (SUBROUTINES GRID AND AREA)

All rays that are traced through the target geometry (Subroutine GRID), or to the first component of the target geometry (Subroutine AREA), originate from the grid plane at the center of the target. The grid plane (refer to examples in Figures 39 and 40) is a plane divided into equal-sized squares called cells and located such that the center of the grid plane is at the origin of the target geometry and oriented in the azimuth and elevation angle. The grid plane is established from the grid size, cell size, and attack angle of the target. The grid size is given in terms of the number of cells (N_x and N_y) in the two dimensions of the grid plane. N_x and N_y should both be odd numbers for best results.

Given the azimuth angle α and the elevation angle θ, the direction cosines of the rays to be fired at the target are

$$\overline{WB}_x = -\cos\theta \cos\alpha \tag{178}$$

$$\overline{WB}_y = -\cos\theta \sin\alpha \tag{179}$$

$$\overline{WB}_z = -\sin\theta \tag{180}$$

where the azimuth and elevation angles are measured in the positive axis direction. An azimuth and elevation angle of 0^o represents a head-on attack plane. An azimuth of 90^o shifts the grid plane to the left flank of the target, and an elevation of 90^o represents an overhead attack plane.

The number of cells in the grid plane is given by

$$N = N_x \cdot N_y$$

By letting I represent the row number and J represent the column number, a specific cell, k, can be located in the grid plane (see example, Figure 40) as follows:

$$I = \text{Integer}\left(\frac{k-1}{N_x}\right) + 1 \tag{181}$$

$$J = k - (I-1)\cdot N_x \tag{182}$$

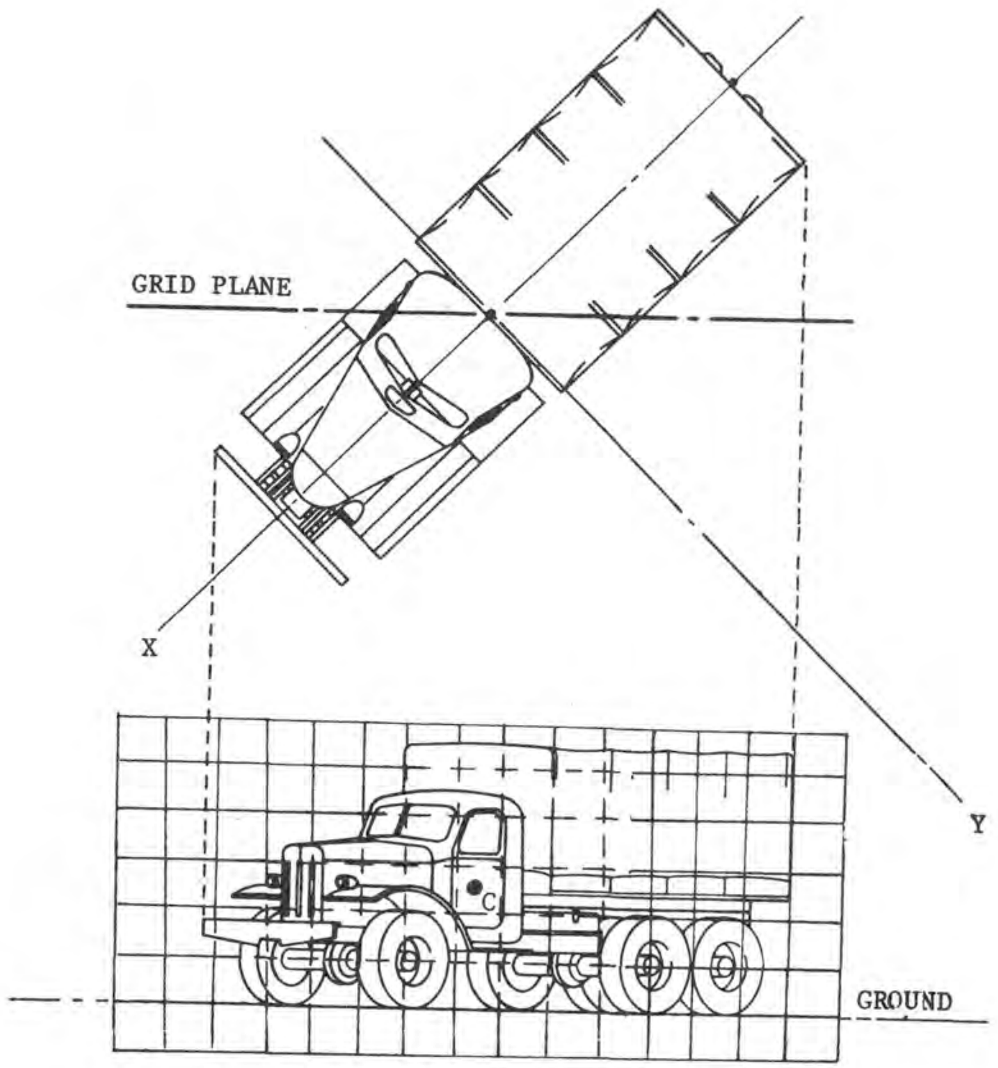
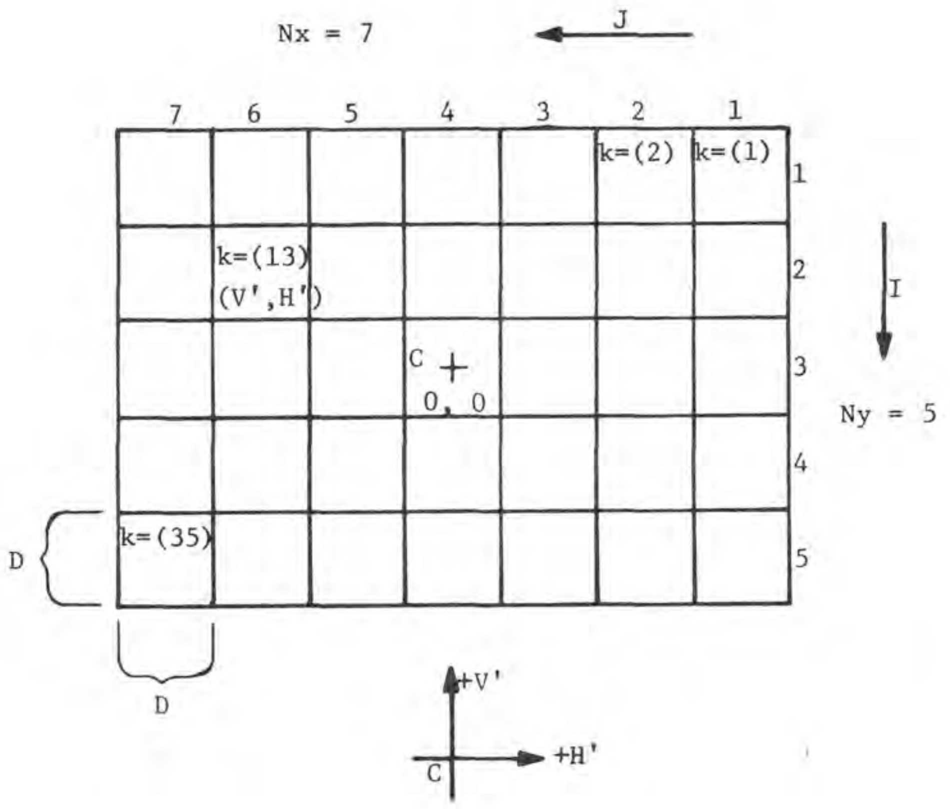


FIG. 39. Grid Plane Geometry



Nx	Number of horizontal cells	C	Center of grid plane (V=0, H=0)
Ny	Number of vertical cells	V'	Vertical distance from center of grid plane to lower left corner of specified grid cell
J	Column number (1 - 7)		
I	Row number (1 - 5)		
k	Specific grid cell number (varies from (1) to (35) in figure above	H'	Horizontal distance from center of grid plane to lower left corner of specified grid cell
D	Dimensions of grid cell		

FIG. 40. Example of a Grid Plane

Therefore, from Figure 40, if cell number 13 were the specific cell desired (k=13), then

$$I = \text{Integer}\left(\frac{13-1}{7}\right) + 1 = 2 \text{ (row 2)} \tag{183}$$

$$J = 13 - (2-1)\cdot 7 = 6 \text{ (column 6)} \tag{184}$$

It should be noted that the row and column numbering begin at the upper right corner of the grid plane.

The location of the lower left corner of the specific grid cell relative to the center, C, of the grid plane is given by

$$V' = [\text{Integer } (N_y/2) - I]\cdot D + D/2 \tag{185}$$

$$H' = [\text{Integer } (N_x/2) - J]\cdot D + D/2 \tag{186}$$

For grid cell number 13 in the example (Figure 40) and assuming a four-inch cell size as in Figures 39 and 40, the coordinates of the lower left corner of grid cell number 13 relative to the center, C, of the grid plane would be

$$V' = [\text{Integer } (5/2) - 2]\cdot 4 + 4/2 = 2$$

$$H' = [\text{Integer } (7/2) - 6]\cdot 4 + 4/2 = -10$$

The mid-point, M, of the specific grid cell can be located for reference purposes by adding one-half the cell size to both V' and H' as follows:

$$V_{ref} = V' + D/2 \tag{187}$$

$$H_{ref} = H' + D/2 \tag{188}$$

From the above example for grid cell number 13 this would be

$$V_{ref} = 2 + 4/2 = 4$$

$$H_{ref} = -10 + 4/2 = -8$$

One of one hundred possible points within the specific grid cell can now be selected as follows:

$$V = V' + D\ (I_v/10) + D/20$$

$$H = H' + D\ (I_h/10) + D/20$$

where I_v and I_h are two random numbers between zero and nine. Since each cell is, in effect, divided into 100 sub-cells, D/20 in the above equations insures that the ray originates from the center of the randomly selected subcell.

If random numbers I_v and I_h are equal to two and six respectively (see example Figure 41), V and H of the above example would be

$$V = 2 + 4(2/10) + 4/20 = 3.0$$

$$H = -10 + 4(6/10) + 4/20 = -7.4$$

for cell number 13 of Figure 40.

The grid plane/cell point coordinates with respect to the center of the grid plane, C, are transformed into the x, y, z coordinates of the target vehicle by

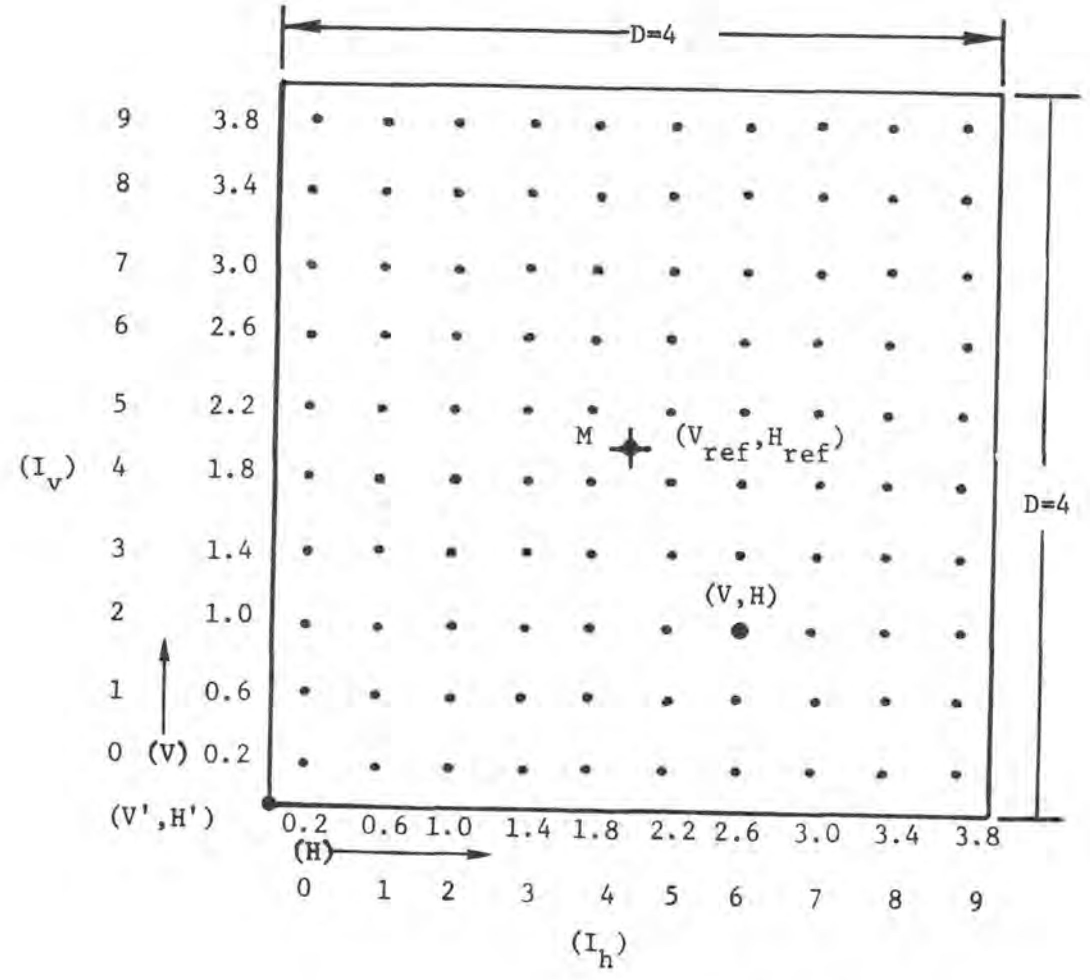
$$\overline{X}_{px} = (XSHIFT) - V \cos \alpha \sin \theta - H \sin \alpha \qquad (189)$$

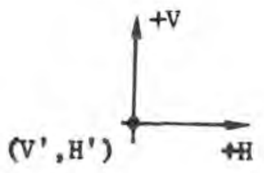
$$\overline{X}_{py} = (YSHIFT) - V \sin \alpha \sin \theta + H \cos \alpha \qquad (190)$$

$$\overline{X}_{pz} = (ZSHIFT) + V \cos \theta \qquad (191)$$

where XSHIFT, YSHIFT, and ZSHIFT are used, in effect, to shift the entire grid plane, if desired. XSHIFT, YSHIFT, and ZSHIFT provide the option of relocating the grid plane to some specific component or components of the target, for example the engine area of the target rather than the entire target or center of the target.

Once the coordinates $\overline{X}_p$ of the point on a given cell have been determined, the point is backed out from the grid plane in the target by a distance ENGTH and direction $\overline{WB}$ to a point $\overline{XB}$. Point $\overline{XB}$ is the origin of the ray to be fired at the target, passing through the grid point before being backed out from the grid plane.





V	vertical distance into grid cell from lower left corner (V',H')
H	horizontal distance into grid cell from lower left corner (V',H')
M	center of grid cell
D	dimensions of grid cell
I_v	random number 0-9 for generating V
I_h	random number 0-9 for generating H

FIG. 41. Example Grid Cell

SOLUTION OF QUARTIC EQUATION (SUBROUTINE QRTIC)

Subroutine QUARTIC computes the roots of a quartic equation of the form

$$x^4 + ax^3 + bx^2 + ex + d = 0 \tag{192}$$

Using the algebraic solution of biquadratic equations discovered by Ferrari, a pupil of Cardon, Equation (192) can be rewritten in the form

$$x^4 + ax^3 = - bx^2 - cx - d \tag{193}$$

Adding $\frac{a^2}{4} x^2$ to both sides of Equation (193) results in

$$\left(x^2 + \frac{a}{2} x\right)^2 = \left(\frac{a^2}{4} - b\right) x^2 - cx - d \tag{194}$$

which is an equation equivalent to the original equation. If the right-hand member of Equation (194) were a perfect square, the solution of the equation would be immediate. However, usually there is not a perfect square. The basic idea in Ferrari's method, therefore, consists in adding to both sides of Equation (194)

$$y \left(x^2 + \frac{a}{2} x\right) + \frac{y^2}{4}$$

to have a perfect square on the left-hand side for an indeterminate y. Equation (194) is then transformed into

$$\left(x^2 + \frac{a}{2} x + \frac{y}{2}\right)^2 = \left(\frac{a^2}{4} - b + y\right) x^2 + \left(\frac{ay}{2} - c\right) x + \left(\frac{y^2}{4} - d\right) \tag{195}$$

y is determined so that the right-hand side of Equation (195) becomes the square of linear expression ex + f.

In general, if

$$Ax^2 + Bx + C = (ex+f)^2 \tag{196}$$

$$= e^2 x^2 + 2efx + f^2$$

then $$B^2 - 4AC = 0 \tag{197}$$

and $$A = e^2,\ B = 2ef,\ C = f^2 \tag{198}$$

so that Equation (197) is satisfied. Therefore, the right-hand side of Equation (195) will be the square of a linear expression cx + f if y satisfies the equation

$$\left(\frac{ay}{2} - c\right)^2 = 4\left(y + \frac{a^2}{4} - b\right)\left(\frac{y^2}{4} - d\right) \tag{199}$$

or, in expanded form,

$$y^3 - by^2 + (ac-4d)y + 4bd - a^2d - c^2 = 0 \tag{200}$$

For y = 2w Equation (200) becomes

$$\boxed{w^3 - \frac{b}{2}w^2 + \frac{(ac-4d)}{4}w + \frac{4bd - a^2d - c^2}{8} = 0} \tag{201}$$

It suffices to take for w any root of this cubic equation, called the resolvent of the biquadratic equation, to result in

$$\boxed{\left(\frac{a^2}{4} - b + 2w\right)x^2 + (aw-c)\,x + (w^2-d) = e^2\,x^2 + 2efx + f^2} \tag{202}$$

with properly chosen e and f. The biquadratic equation therefore appears in the form

$$\left(x^2 + \frac{a}{2} + \frac{1}{2}y\right)^2 = (ex + f)^2$$

which splits into two quadratic equations

$$x^2 + \frac{a}{2}x + \frac{1}{2}y = \pm(ex + f)$$

Solving for x using the quadratic formula results in the four requested roots

$$\boxed{x_{1,2} = \frac{-\frac{a}{2} + e}{2} \pm \frac{\sqrt{\left(-\frac{a}{2} + e\right)^2 - 4(w-f)}}{2}} \tag{203}$$

and

$$\boxed{x_{3,4} = \frac{-\frac{a}{2} - e}{2} \pm \frac{\sqrt{\left(-\frac{a}{2} - e\right)^2 - 4(w+f)}}{2}} \tag{204}$$

SOLUTION OF CUBIC EQUATION (SUBROUTINE CUBIC)

Subroutine CUBIC computes the roots of a cubic equation of the form

$$x^3 + ax^2 + bx + c = 0 \tag{205}$$

Using Cardan's solution, Cubic Equation (205) is transformed to the reduced form

$$y^3 + py + q = 0 \tag{206}$$

where

$$\boxed{p = b - \frac{a^2}{3}} \tag{207}$$

and

$$\boxed{q = c+a\left(2\,\frac{a^2}{27} - \frac{b}{3}\right)} \tag{208}$$

through the substitution of

$$x = y - \frac{a}{3} \tag{209}$$

The roots y_1, y_2, and y_3 of reduced Cubic Equation (206) are

$$\boxed{y_1 = A + B} \tag{210}$$

$$\boxed{y_{2,3} = - \frac{A + B}{2} \pm i\,\frac{A - B}{2}\sqrt{3}} \tag{211}$$

with

$$\boxed{A = \sqrt[3]{-\frac{q}{2} + \sqrt{Q}}} \qquad (212)$$

$$\boxed{B = \sqrt[3]{-\frac{q}{2} - \sqrt{Q}}} \qquad (213)$$

$$\boxed{Q = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^2} \qquad (214)$$

where the real values of the cube roots are used. The Cubic Equation has one real root and two conjugate complex roots; three real roots of which at least two are equal; or three different real roots, if Q is respectively positive, zero, or negative. If Q of Equation (214) is negative, y is solved by the trigonometric solution

$$\boxed{y_1 = 2\sqrt{-p/3}\,\cos\,(\phi/3)} \qquad (215)$$

and

$$\boxed{y_{2,3} = -2\sqrt{-p/3}\,\cos\,(\phi/3 + 60^\circ)} \qquad (216)$$

where

$$\boxed{\cos\,\phi = -\frac{q}{2\sqrt{-(p/3)^3}}} \qquad (217)$$

The real roots of x in Equation (205) are determined by substituting the real values of y into Equation (209).

THE MASTER-ASTER ARRAY

The MASTER-ASTER array is a 10,000-word array used in the MAGIC program to store the processed target geometry data. In its completed form (after Subroutine AREA) the MASTER-ASTER array contains the following information. (The variable name in parentheses represents the location of the first word of data of each section.)

1. RPP pointer data (LBASE)
    a. pointer to abutting RPP's for given side
    b. number of abutting RPP's for given side
    c. pointer to coordinate for given side of RPP
2. RPP coordinate data (LRPPD)
3. List of abutting RPP's for each side of each RPP
4. Body pointers (LABUT)
    a. body type
    b. pointer to body data pointers
    c. pointer to region enter table
    d. pointer to region leave table
    e. number of regions in enter table
    f. number of regions in leave table
5. Body data pointers (LBODY)
6. Body data pointers followed by body dimensions (LBOD)
7. Region data pointers (LREGD)
    a. pointer to operator/body list for given region
    b. number of bodies in list for given region
8. Region data (LREGL)
    a. operator
    b. body

9. Region enter/leave tables (LENLV)

10. Body enter intersect distances (LRIN)

11. Body exit intersect distances (LROT)

12. Subroutine G1 working storage (LIO)

    a. entering surface number

    b. exit surface number

    c. internal loop counter

13. Presented area by component code (LAREA)

14. Region code data (LIRFO)

    a. component identification data

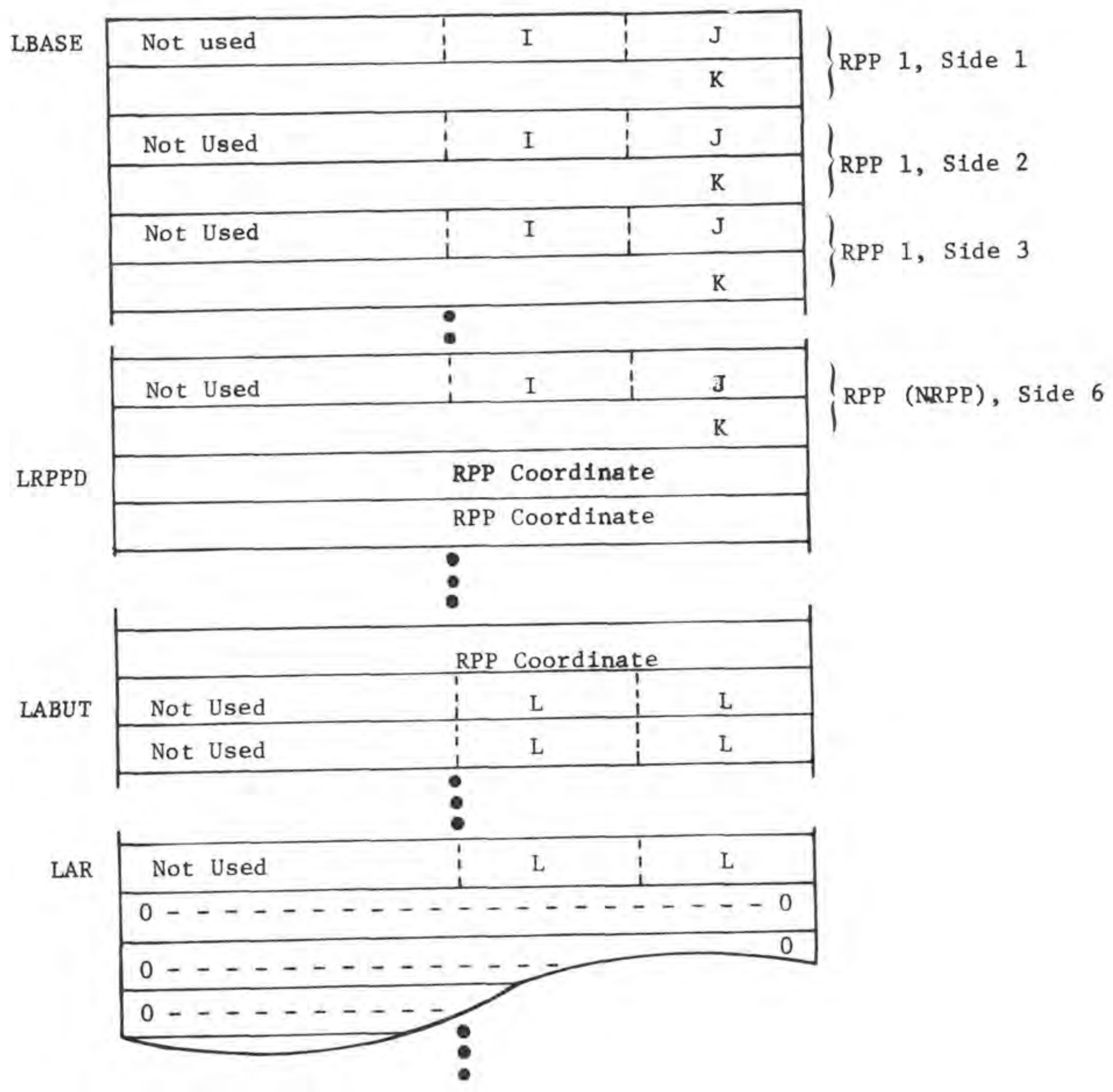
    b. space identification data

15. Last word of the MASTER-ASTER array (NDQ)

Data for the MASTER-ASTER array is entered throughout the execution of the entire MAGIC program. Therefore, to gain a complete understanding of the contents of the MASTER-ASTER array at various times in the execution of the MAGIC program, the following discussion, with accompanying figures, is given.

The MAIN program assigns the first available storage location as the first word of the MASTER-ASTER array. This location is assigned the variable name LBASE. The last word of the MASTER-ASTER array, word 10,000, is assigned the variable name NDQ. Before Subroutine GENI is called, the entire MASTER-ASTER array is cleared (zeroed).

When Subroutine GENI is called, the first data to be entered into the MASTER-ASTER array is the RPP geometry. To accomplish this, Subroutine GENI calls Subroutine RPPIN to enter and process the RPP data. When Subroutine RPPIN returns control to Subroutine GENI, the contents of the MASTER-ASTER array are as shown in Figure 42.



(continue with Figure 43)

FIG. 42. RPP Data Storage (Subroutine RPPIN)

LBASE, starting location (equal to one) of RPP pointer section (12 words/RPP)

I, pointer to list of abutting RPP's for given side (15 bits)

J, number of RPP's that abut given side (15 bits)

K, pointer to RPP coordinate for given side

LRPPD, RPP x, y, or z, maximum or minimum coordinate value. The location of LRPPD is equal to LBASE + 12*NRPP. Pointer K in section LBASE points to a specific RPP coordinate in this section

LABUT, lists of abutting RPP's to given surfaces, packed one or two per word. Pointer I in section LBASE points to a specific list of abutting RPP's in this section

L, abutting RPP's (15 bits)

LAR, last location of RPP data

FIG. 42. RPP Data Storage (Subroutine RPPIN) (Concluded)

After all RPP data has been entered into the MASTER-ASTER array by Subroutine RPPIN, the body data is entered, processed, and stored in the MASTER-ASTER array. The new body dimensions are stored backward from location NDQ with pointer LBOT always equal to the new starting location of the body dimensions. The individual body dimensions are located by pointers and computed and stored in the LBOD section of the MASTER-ASTER array. The overall pointer to a group of pointers for a given body is stored in the LBODY section following the RPP data section as shown in Figure 43.

When all body data is processed and stored in the MASTER-ASTER array, there exists a large unused section in the array between the LBOD body dimension pointers and the body dimensions at the end of the MASTER-ASTER array at location LBOT through NDQ (see Figure 43). Therefore, Subroutine GENI shifts all the data at the end of the MASTER-ASTER array, LBOT through NDQ, to the next available location (LDATA) after the LBOD body dimension pointer section. The body dimension pointers in section LBOD are also adjusted to account for the shift of the body dimensions. When the shift is complete, LDATA points to the next available location in the MASTER-ASTER array. It should also be noted that due to the shift of LBOT through NDQ, pointer LBOT is no longer correct and the variable name is dropped.

After all body data is entered, processed, and stored, and the body dimensions at the end of the MASTER-ASTER array have been shifted, the region data is entered and stored following the shifted body dimension data as shown in Figure 44.

When the region data has all been entered and stored, the region enter/leave tables are prepared for each body in the target geometry, and are stored following the region data in the MASTER-ASTER array. During preparation and storing of these tables, the remaining data (LE, LL, NE, NL) for each body in the LBODY section is computed and stored (see Figure 43). The region leaving table for each body is prepared and stored first, immediately followed by the region entering table for each body as shown in Figure 45.

After the region enter/leave tables have been prepared and stored, storage space in the MASTER-ASTER array is reserved for the entry intersect distance data (LRIN), the exit intersect distance data (LROT), and Subroutine G1 working storage (LIO). Data for these three reserved storage sections is computed during Subroutine G1. The next available storage location after the LIO section, LEGEOM, is also determined. One word per body is reserved for each of the sections LRIN, LROT, and LIO as shown in Figure 46.

(Refer to Figure 42)

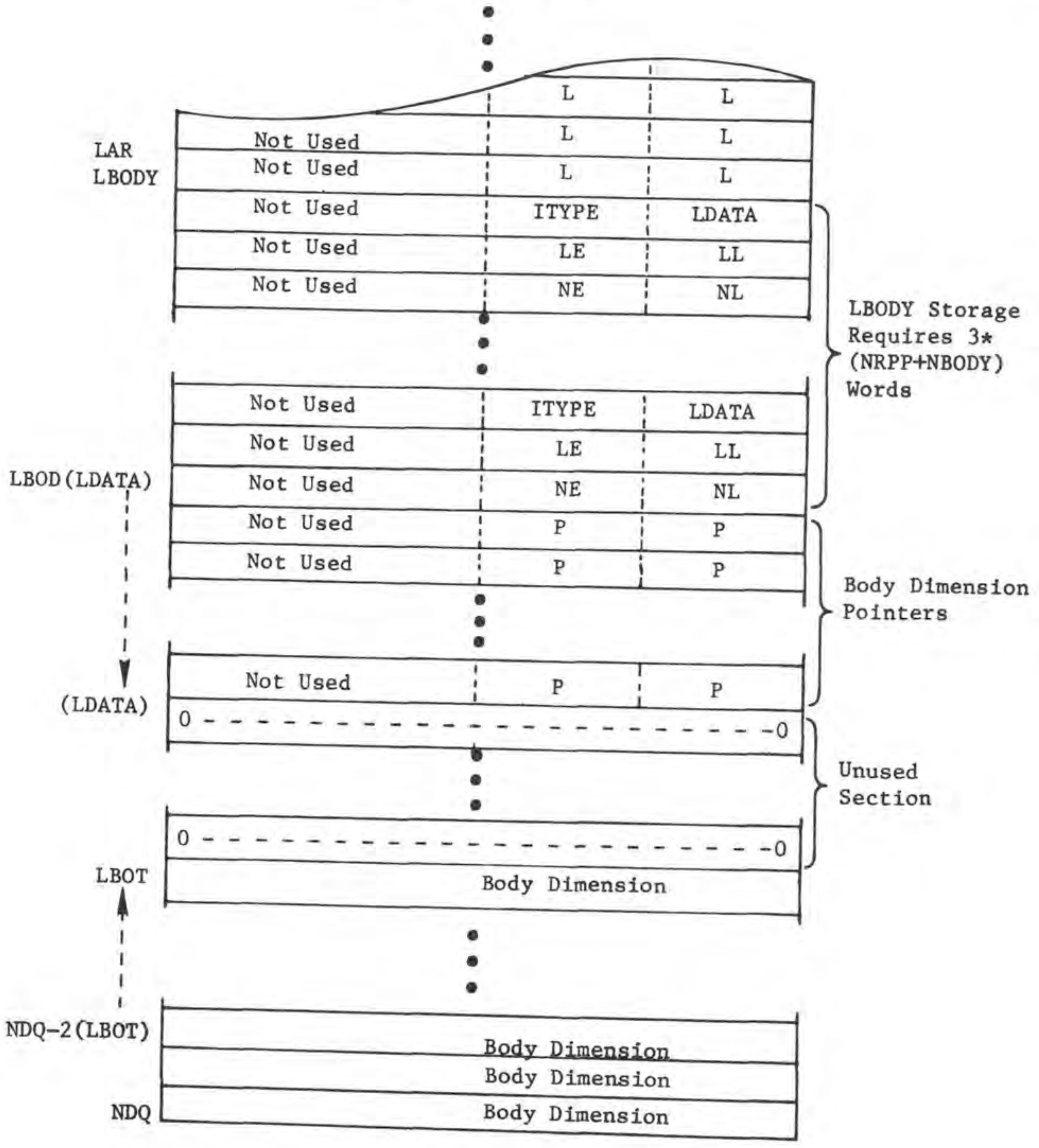


FIG. 43. Body Pointer Storage (Subroutine GENI)

LBODY, starting location of body pointer storage. The location of LBODY is equal to the first location following the RPP data section.

ITYPE, body type (15 bits)

LDATA, pointer for storing pointers in LBOD section for given body (15 bits). When all pointers are stored, LDATA is equal to the next available location after the LBOD section.

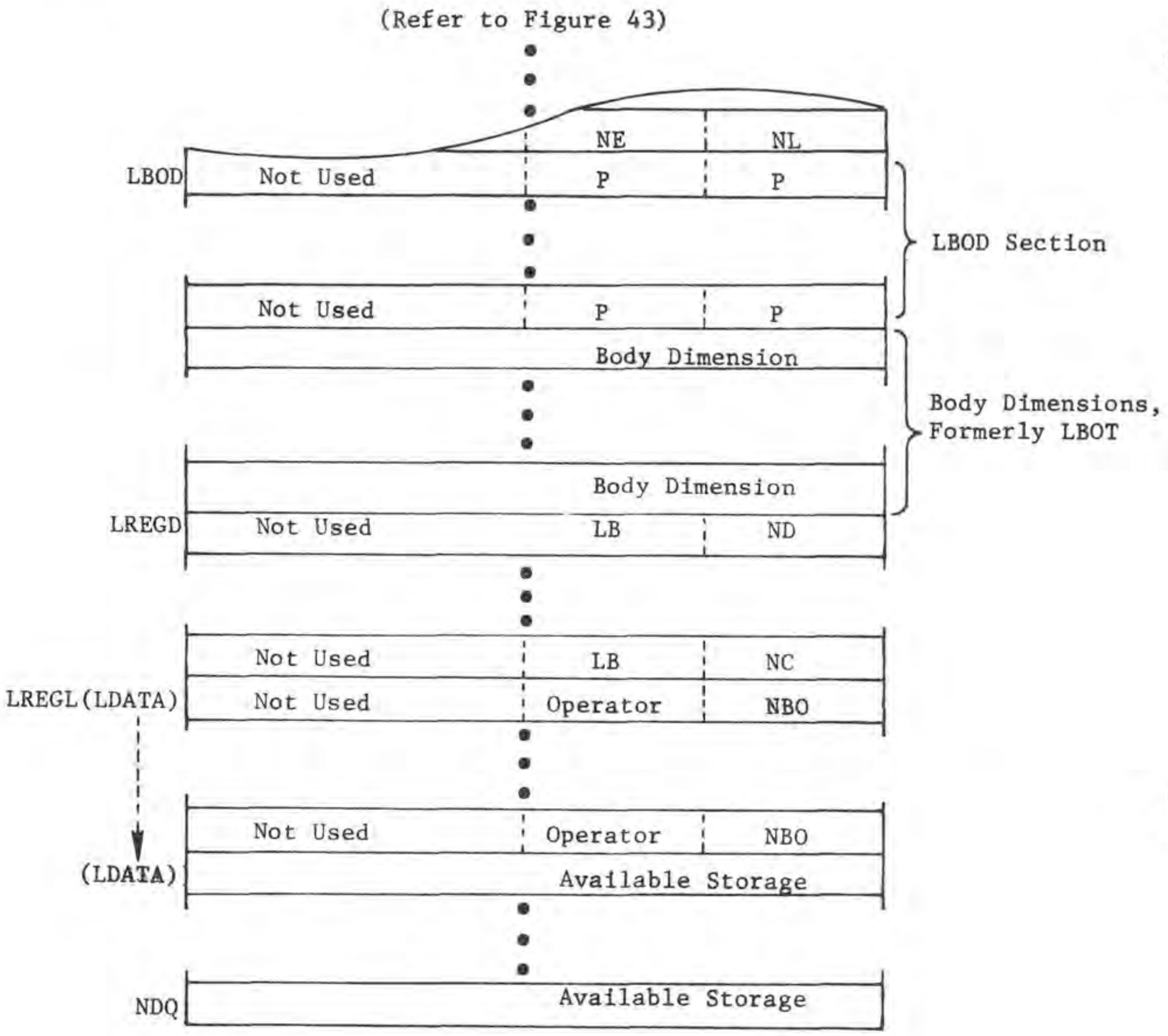
LE, LL, NE, NL, nothing stored in these locations at this time in program (contain zeroes). For eventual contents, refer to the discussion of Figure 45.

LBOD, starting location of pointers to body dimensions in section LBOT.

P, pointers to body dimensions. Packed two per word (15 bits). LDATA of LBODY section points to specific pointers required for a given body

LBOT, starting location of body dimensions.

FIG. 43. Body Pointer Storage (Subroutine GENI) (Concluded)



LREGD, starting location of region pointer data

LB, pointers to operator/body list for a given region (15 bits)

NC, number of bodies in list for a given region (15 bits)

LREGL, lists of operator/body for each region. LB of LREGD points to a specific list in this section

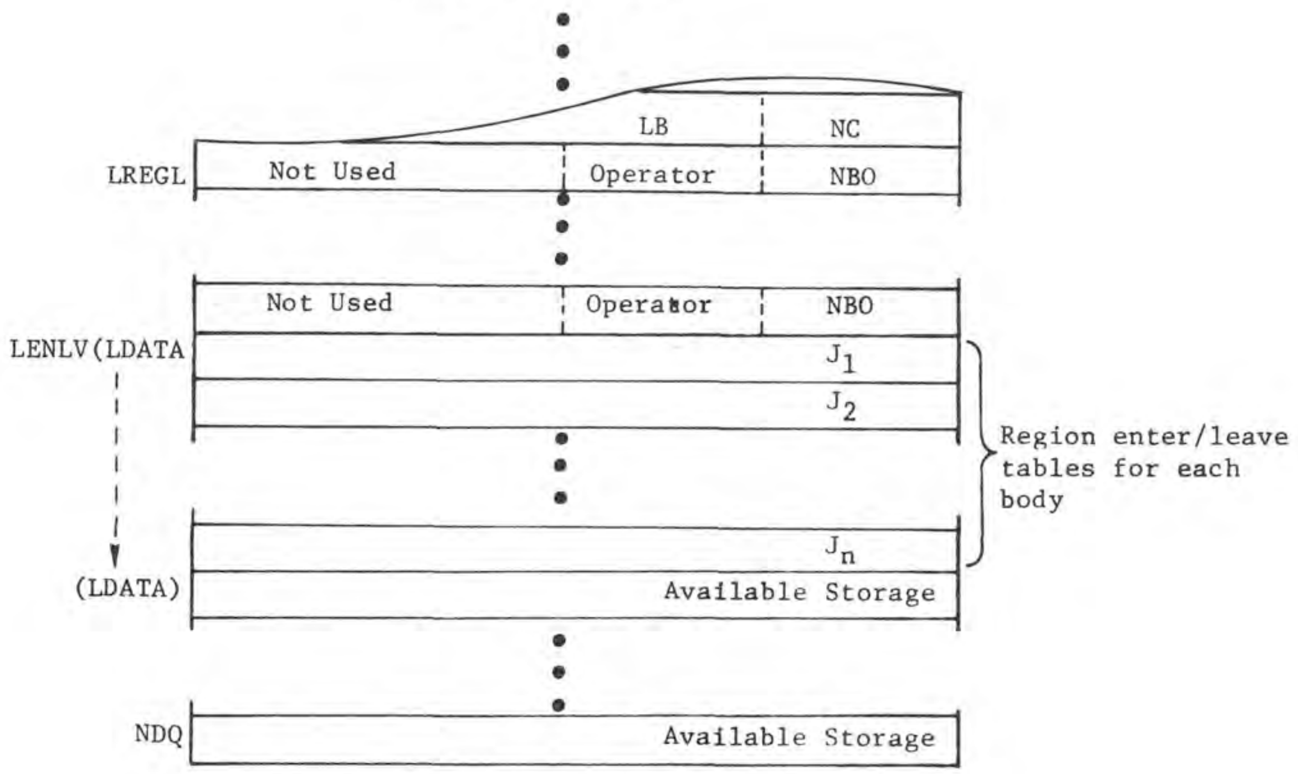
Operator-(OR), (+), or (-) to denote relationship of given body to region (15 bits)

NBO, body number of specific body in region (15 bits)

(LDATA), pointer for storing operator/body in LREGL region data section in next available storage location. When all region data are entered, LDATA points to next available storage location after the region data.

FIG. 44. Region Storage (Subroutine GENI)

(Refer to Figure 44)



LENLV, starting location of the region enter/leave tables for each body

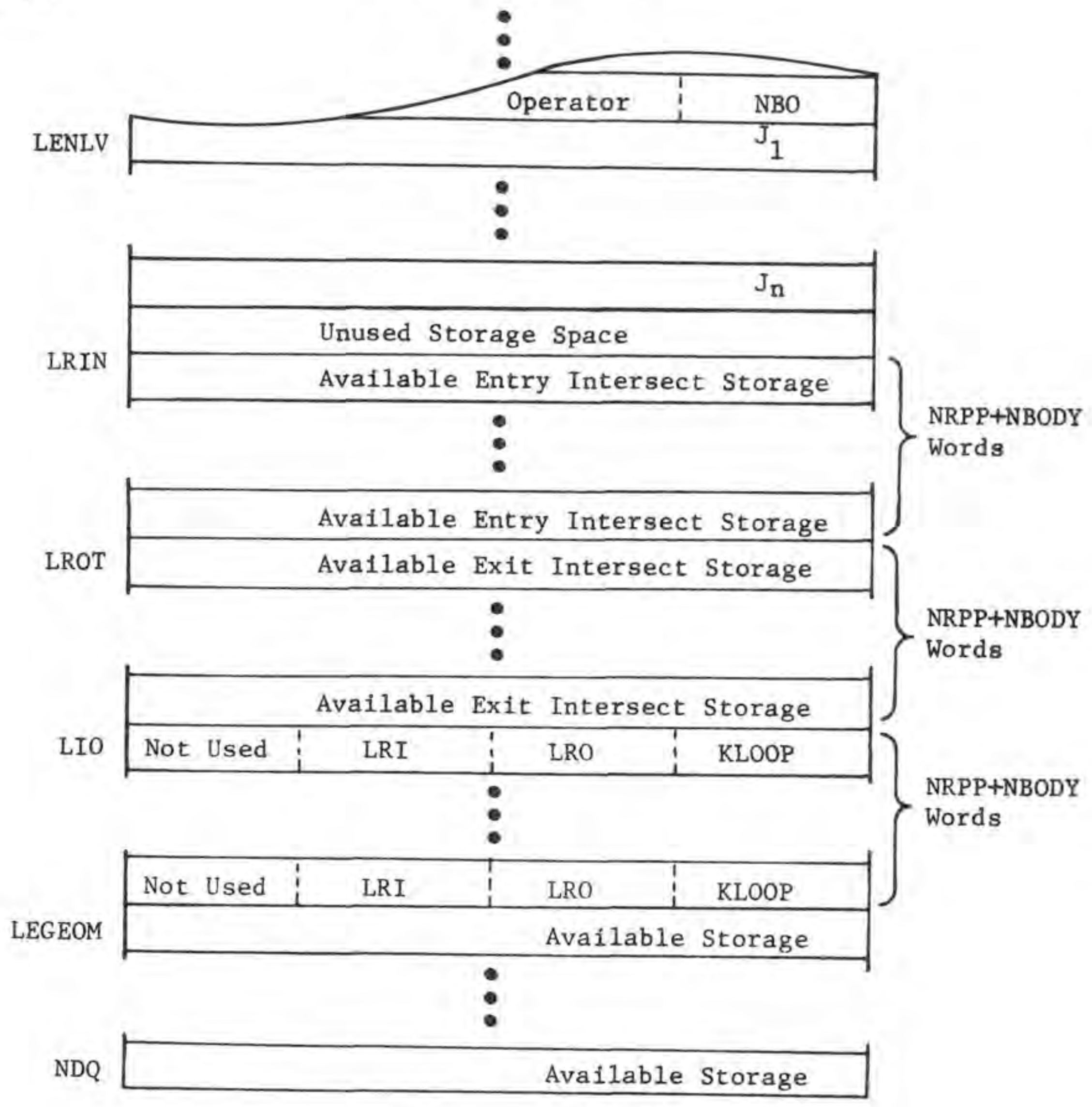
J, first region entry for first body of first leaving table

Jn, last region entry for last body of last entering table

(LDATA), pointer for storing regions in enter/leave tables at next available storage location. When all enter/leave tables are entered, LDATA points to the next available storage location after the enter/leave tables.

FIG. 45. Region Enter/Leave Tables (Subroutine GENI)

(Refer to Figure 45)



LRIN, starting location reserved for entry intersect distance data. One storage word is reserved for each body in the target geometry

LROT, starting location reserved for exit intersect distance data. One storage word is reserved for each body in the target geometry

LIO, starting location reserved for Subroutine G1 for working storage. One storage word is reserved for each body in the target geometry

LRI, body surface number of entry intersect (6 bits)

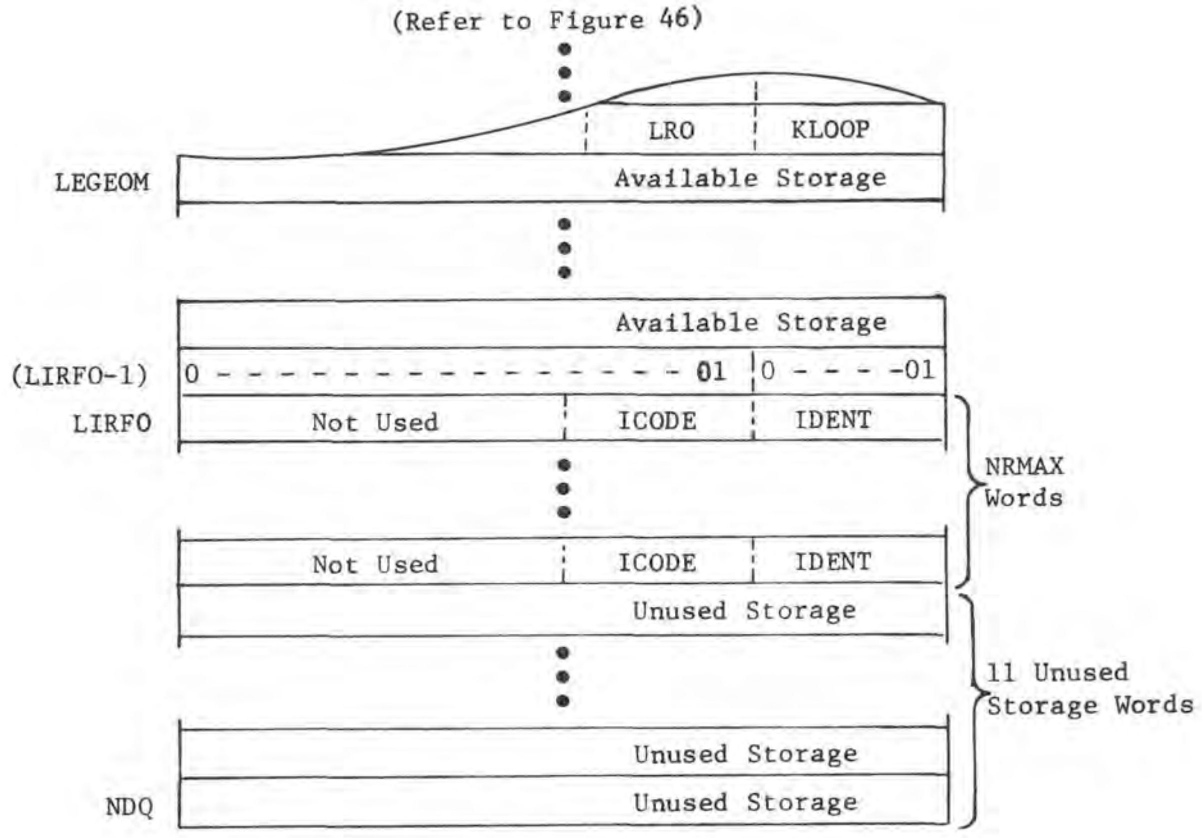
LRO, body surface number of exit intersect (6 bits)

KLOOP, internal loop counter (15 bits)

LEGEOM, the next available location at the end of the target geometry

FIG. 46. Reserved Storage (Subroutine GENI)

After the LRIN, LROT, and LIO storage space has been reserved in the MASTER-ASTER array, control is returned to the MAIN program where the next phase relative to the MASTER-ASTER array is the entry and storage of the region component and identification code data; it is stored near the end of the MASTER-ASTER array and preceded by a special code as shown in Figure 47).



LIRFO, starting location of region component/identification code data. The location of LIFRO is equal to NDQ-NRMAX-10

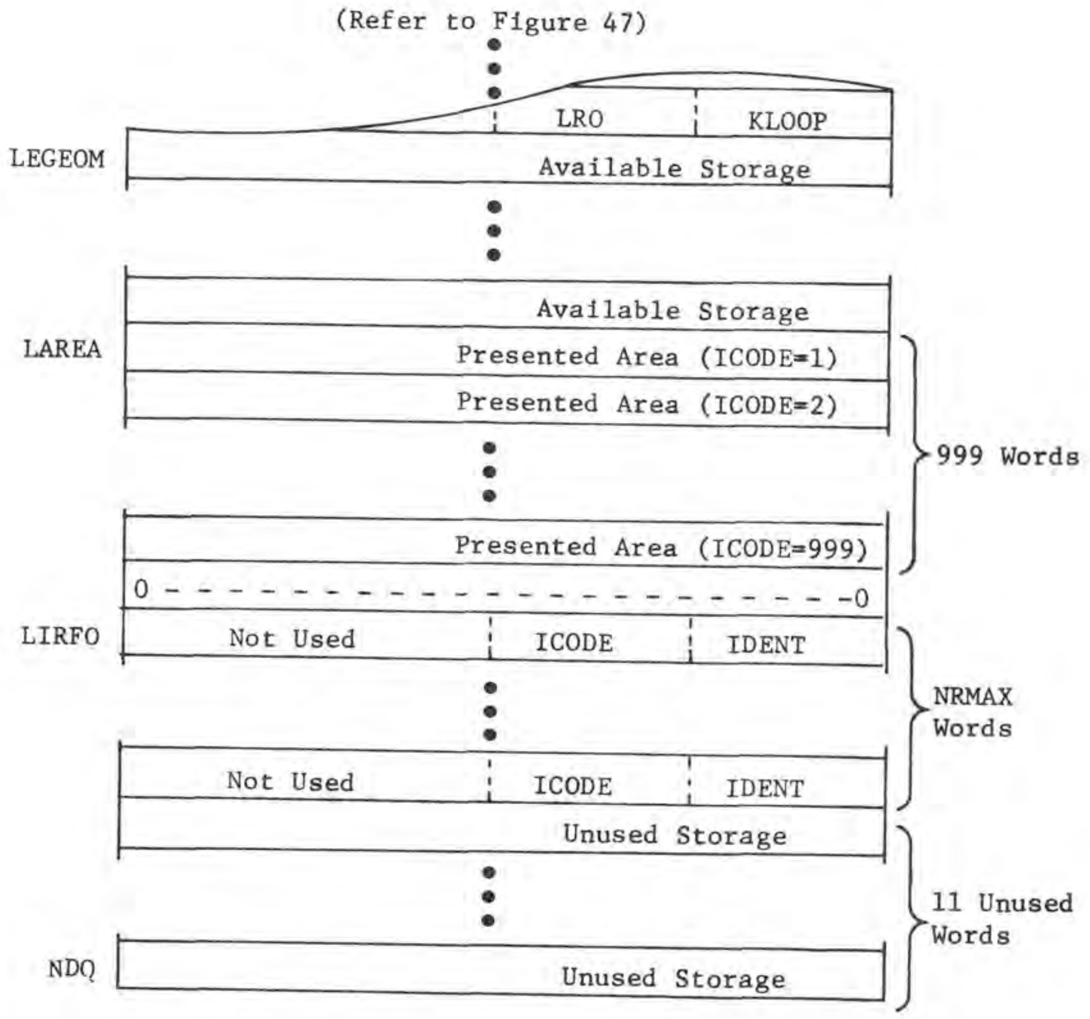
ICODE, region component code (15 bits)

IDENT, region identification code. Stored as IDENT+1 to prevent storing a negative number (15 bits)

(LIRFO-1), special code that precedes region component/identification code data. Not used in MAGIC program.

FIG. 47. Region Component/Identification Code Data (MAIN Program)

If Subroutine AREA is to be executed, the last data to be stored in the MASTER-ASTER array is the presented area of the target by component code (1-999). The presented area data is stored in the MASTER-ASTER array preceding the region component/identification code data, as shown in Figure 48. One-thousand storage locations are reserved for the presented area of the target by component code (1-999). The 1000 storage locations are zeroed before the presented areas are stored. Therefore, the special code at location LIRFO-1 explained above is also zeroed.



LAREA, starting location of the presented areas by component code of the target from a given azimuth and elevation angle. The location of LAREA is equal to LIRFO-1000.

FIG. 48. Presented Area by ICODE (Subroutine AREA)

PROGRAM MAGIC (MAIN PROGRAM)

The purpose of the MAIN routine of the MAGIC program is to direct the entire flow of the MAGIC program, calling on various subprograms to perform specific functions.

There are essentially two phases to the MAGIC program: geometry input processing and main ray-tracing control. The MAIN routine initializes various program constants and parameters and enters from card input the program control parameters. The target geometry is entered by Subroutine GENI, which enters and prepares the target geometry data for the ray-tracing phase of the MAGIC program. The target geometry can then be written out on tape. If the target geometry data has already been processed by a previous run, the input tape is entered, and Subroutine GENI is therefore not called.

After the input processing phase, the MAIN program has the option, based on the input control parameters, of calling Subroutine TESTG to fire a ray from point to point and of calling Subroutine VOLUM to compute the volumes by region. The region identifiers and space code numbers are then entered and stored into the MASTER-ASTER array.

At this point, the ray-tracing phase of the program is initiated. A card is entered with the number of angles of attack, and Subroutine GRID is called to perform the ray-tracing control phase for each aspect angle. When the ray-tracing phase of the program is complete, the MAIN program has the option, based on the input control parameter, of calling Subroutine AREA to compute the presented area by component for each aspect angle.

The following description is an outline of the purpose and flow of the MAIN program. The steps that describe the program follow the steps of the conceptual flowchart of Figure 49.

Step 1

Initialize various constants and data to be used in the program, and enter and set the options to be used in the program.

Step 2

Test to determine if the target description data is to be entered from a tape prepared by a previous run of the target. If it is, the tape is read in and the title of the target geometry is printed out.

Step 3

If the target description is not to be entered from tape, the entire ASTER array is zeroed.

Step 4

Call Subroutine GENI to enter, process, and organize the target geometry input data into a usable format for use by subsequent subroutines.

Step 5

After the target description data has been entered and processed by Subroutine GENI, the option for putting the target description on tape for subsequent runs is tested. If positive, the target geometry is written out on tape.

Step 6

Test the Subroutine TESTG option to determine if Subroutine TESTG is to be called to fire and trace a ray from one given point to another.

Step 7

Test the Subroutine VOLUM option to determine if it is to be called to compute the volumns of a region(s) of the target.

Step 8

Enter, print out, and store for each region of the target geometry the component code, space code, and special identification. After all component code and identification code data for the regions have been entered, enter another program option card.

Step 9

Test the Subroutine GRID option to determine if Subroutine GRID is to be called to generate and control all of the input and processing for a single attack plane.

Step 10

Test the Subroutine AREA option to determine if Subroutine AREA is to be called to compute the presented area of the target from the given attack plane. The program then terminates.

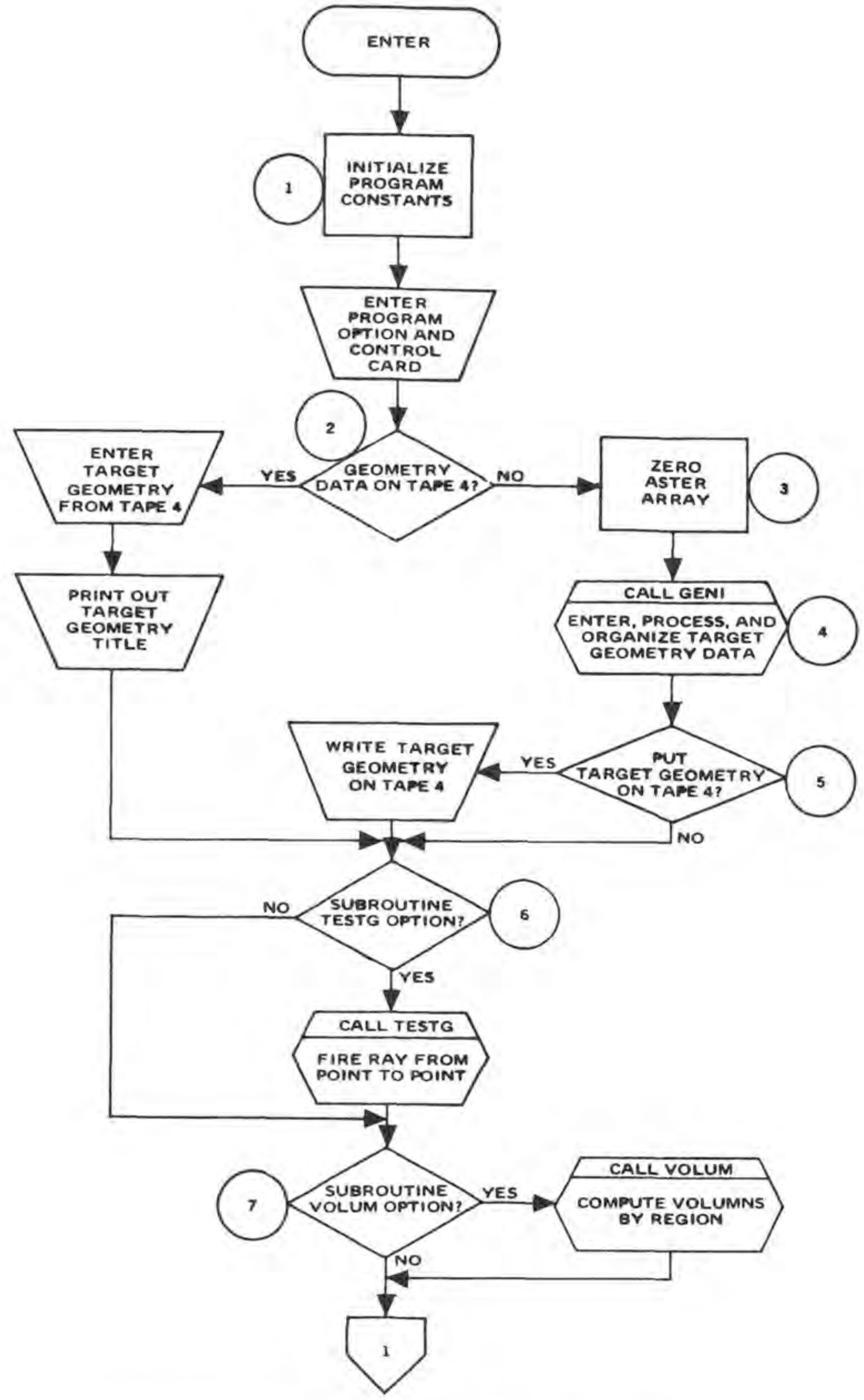

ENTER
1
INITIALIZE PROGRAM CONSTANTS
ENTER PROGRAM OPTION AND CONTROL CARD
2
GEOMETRY DATA ON TAPE 4?
YES
NO
ENTER TARGET GEOMETRY FROM TAPE 4
PRINT OUT TARGET GEOMETRY TITLE
ZERO ASTER ARRAY
3
CALL GENI
ENTER, PROCESS, AND ORGANIZE TARGET GEOMETRY DATA
4
PUT TARGET GEOMETRY ON TAPE 4?
5
YES
NO
WRITE TARGET GEOMETRY ON TAPE 4
SUBROUTINE TESTG OPTION?
6
NO
YES
CALL TESTG
FIRE RAY FROM POINT TO POINT
SUBROUTINE VOLUM OPTION?
7
YES
NO
CALL VOLUM
COMPUTE VOLUMNS BY REGION
1

FIG. 49. MAIN Program Concept Flowchart

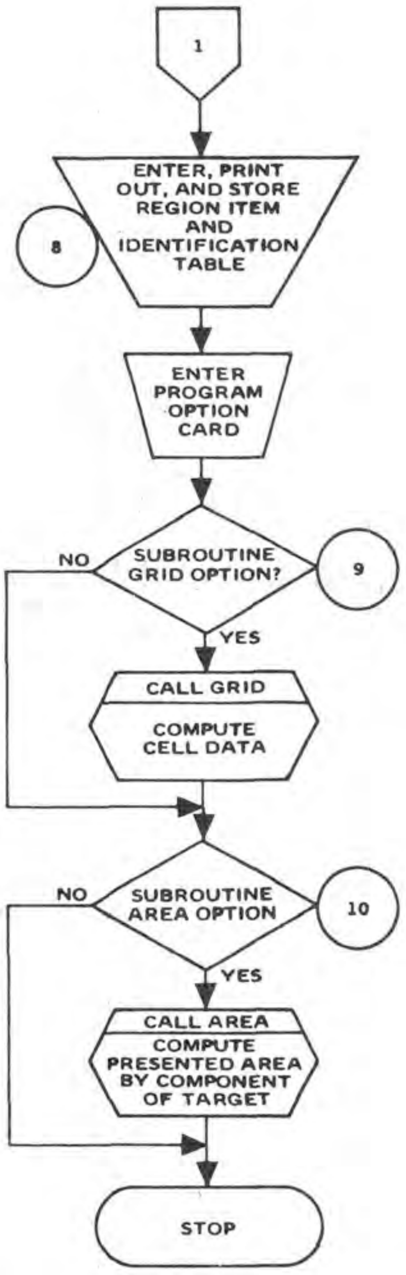

1
ENTER, PRINT OUT, AND STORE REGION ITEM AND IDENTIFICATION TABLE
8
ENTER PROGRAM OPTION CARD
SUBROUTINE GRID OPTION?
NO
YES
9
CALL GRID
COMPUTE CELL DATA
SUBROUTINE AREA OPTION
NO
YES
10
CALL AREA
COMPUTE PRESENTED AREA BY COMPONENT OF TARGET
STOP

FIG. 49. MAIN Program Concept Flowchart (Concluded)

INPUT PROCESSING (SUBROUTINE GENI)

Subroutine GENI is the main control routine for target geometry input processing. Its purpose is to organize the target geometry input data into a usable format for subsequent subroutines. All of the organized data is stored in a large array called the MASTER-ASTER array.

The processing of the body input descriptions is accomplished by calling Subroutine RPPIN to enter and process the RPP data, calling Subroutine ALBERT to enter and process the arbitrary polyhedron (ARB) data, or calling Subroutine ARIN to enter and process the arbitrary surface (ARS) data. All other bodies are entered and processed directly by Subroutine GENI.

When all of the body input data has been processed, Subroutine GENI enters the input data for the region descriptions, stores it in the MASTER-ASTER array, and then computes the region enter/leave tables.

The following description is an outline of the purpose and flow of Subroutine GENI. The steps that describe Subroutine GENI follow the steps of the conceptual flowchart of Figure 50.

Step 1

The body counter array is cleared for the new input data. This array counts the number of times each body type is used to describe the target geometry.

Step 2

The title for the current target geometry is entered and printed out.

Step 3

Enter and print program control parameters for the number of RPP's, the number of bodies, the number of regions, the MASTER-ASTER array print option, and the test region data option.

Step 4

Test to determine if there are to be any RPP's in the target geometry. If so, branch to call the RPP processing subroutine.

Step 5

Call Subroutine RPPIN if an RPP(s) is used to describe the target geometry to enter, process, check, print, and store the RPP data in the MASTER-ASTER array.

Step 6

Compute and assign locations for storing body data pointers and body dimension data.

Step 7

Enter body data card containing program control parameters, body type, and six body dimensions for the given body. This is the beginning of a loop used to enter and process all of the body data of the target geometry.

Step 8

Increment the counter for the specific body type entered and compute and store pointer for the body data.

Step 9

Test body type and branch accordingly to section for entering additional data if required and to compute, process, and check body data.

Step 10

If the body type is a BOX, RCC, REC, TRC, ELL, RAW, TEC, or TOR, set number for additional body dimensions to be entered for the given body.

Step 11

Enter the number of additional body dimensions as determined from Step 10 and print all of the body data for the given body.

Step 12

Compute additional data that is required for the body that was not part of the input and perform check on the data. For a complete and detailed discussion of the computation and testing of the body data, refer to the appropriate section from program statement 250 to statement 300 of the simulation model.

Step 13

If the body type is a sphere, the body dimensions are all on one data card so no additional input is required and the input is printed out.

Step 14

Call Subroutine SEE3 to store the body data and body data pointers for a SPH, BOX, RCC, REC, TRC, ELL, RAW, TEC, or TOR in the MASTER-ASTER array.

Step 15

If the body type is an arbitrary surface (ARB), the data entered by the initial card is printed out.

Step 16

Call Subroutine ALBERT to enter additional ARB data and to compute additional data required. The data and pointers are then stored in the MASTER-ASTER array by the subroutine.

Step 17

Call Subroutine ARIN to enter, check, process, and store data for the arbitrary surface (ARS).

Step 18

Test to determine if more body data is to be entered for the target geometry and if so, return to enter the next body data/control card.

Step 19

Print out a table giving the number of times each body was used in the target description, and print out the starting locations of major data/pointer sections in the MASTER-ASTER array.

Step 20

When the body data is first entered, it is stored toward the end of the MASTER-ASTER array with the pointer data and RPP data at the front. This step moves the body data to just after the pointer data.

Step 21

Initialize counters, and assign and compute pointers for entering, processing, and checking region combination data.

Step 22

Enter region data card containing region number, the logical operator, and all of the bodies that comprise the region.

Step 23

Check validity of the region data by testing each body number in the region description to verify that each body number is not greater than the number of bodies plus the number of RPP's used to describe the target geometry.

Step 24

Print out the region data, and compute the pointer to the region data for the region and store the region data.

Step 25

Test for logical operator and convert to integer depending upon whether operator was positive or negative to represent an AND or OR condition.

Step 26

Pack and store the operator and body number in the region data section and continue testing all operators and body numbers in the given region.

Step 27

Branch to enter data for next region if all regions have not been entered and processed.

Step 28

Verify that all regions of the target geometry have been entered and that there were no invalid body numbers in the region data.

Step 29

Test program input control parameter to determine if the validity of the region data is to be checked.

Step 30

If the validity of the region data is to be checked, each of the operators and body numbers of each region is compared with each of the operators and body numbers of all other regions to verify that there is no area in one region that is identical to another region. For a complete and detailed discussion of the testing of the region data, refer to the DO 456 DO loop (in the simulation model) of Subroutine GENI.

Step 31

Compute and store the region enter/leave tables. For a complete and detailed discussion of the method used in preparing and storing these tables, refer to the DO 590 DO loop in the simulation model of Subroutine GENI.

Step 32

Reserve one word of storage for each geometric shape used to describe the target geometry for the entry intersect data, the exit intersect data, and the Subroutine G1 temporary storage area, and assign pointers to these areas plus region data pointers.

Step 33

Test program input control parameter to determine if the region enter/leave tables are to be printed. If not, control is returned to the MAIN program.

Step 34

Print out region enter/leave tables.

Step 35

Test program input control parameter to determine if the MASTER-ASTER array to the end of the region enter/leave tables is to be printed out. If not, control is returned to the MAIN program.

Step 36

Print out the MASTER-ASTER array to the end of the region enter/leave tables and return control to the MAIN program.

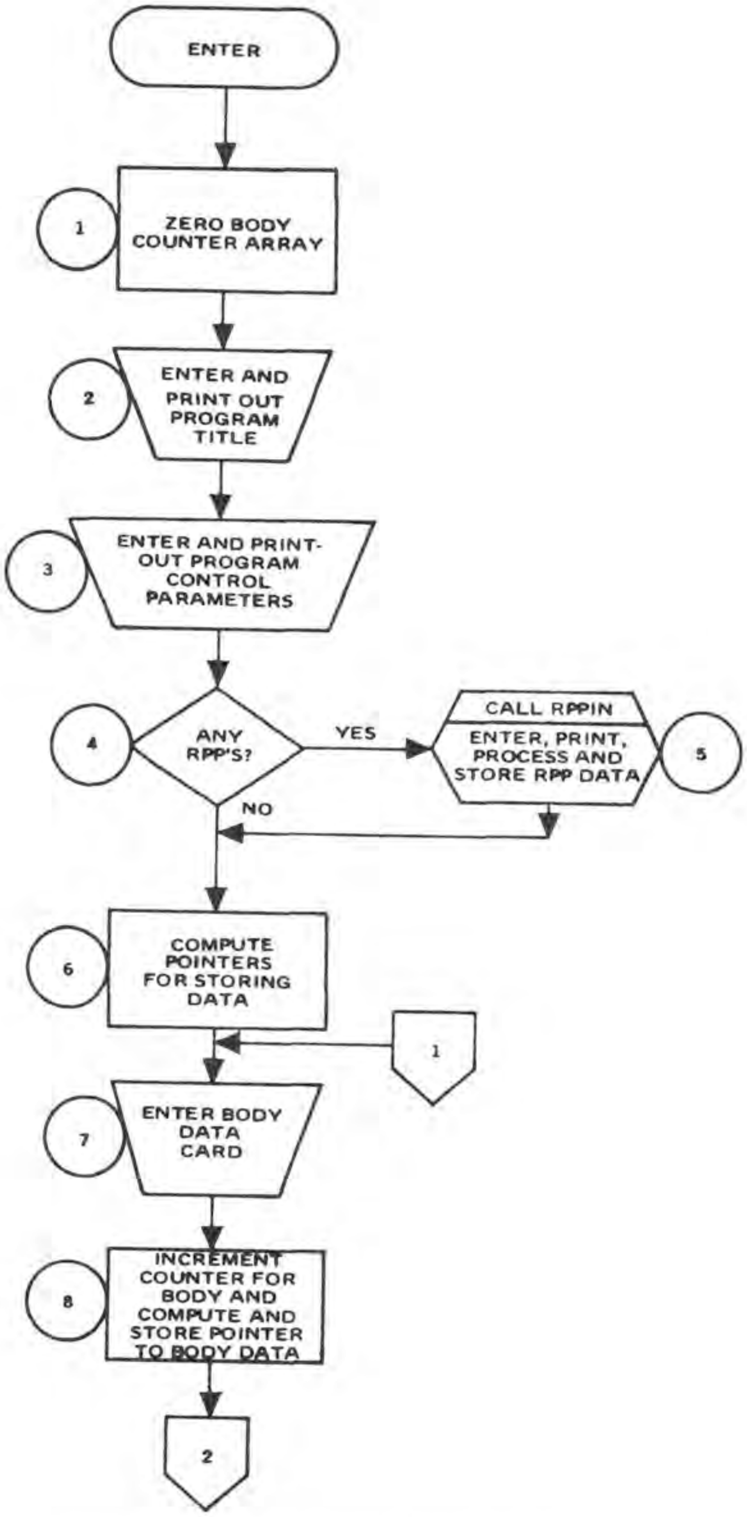

ENTER
1
ZERO BODY COUNTER ARRAY
2
ENTER AND PRINT OUT PROGRAM TITLE
3
ENTER AND PRINT-OUT PROGRAM CONTROL PARAMETERS
4
ANY RPP'S?
YES
NO
CALL RPPIN
ENTER, PRINT, PROCESS AND STORE RPP DATA
5
6
COMPUTE POINTERS FOR STORING DATA
1
7
ENTER BODY DATA CARD
8
INCREMENT COUNTER FOR BODY AND COMPUTE AND STORE POINTER TO BODY DATA
2

FIG. 50. Subroutine GENI Concept Flowchart

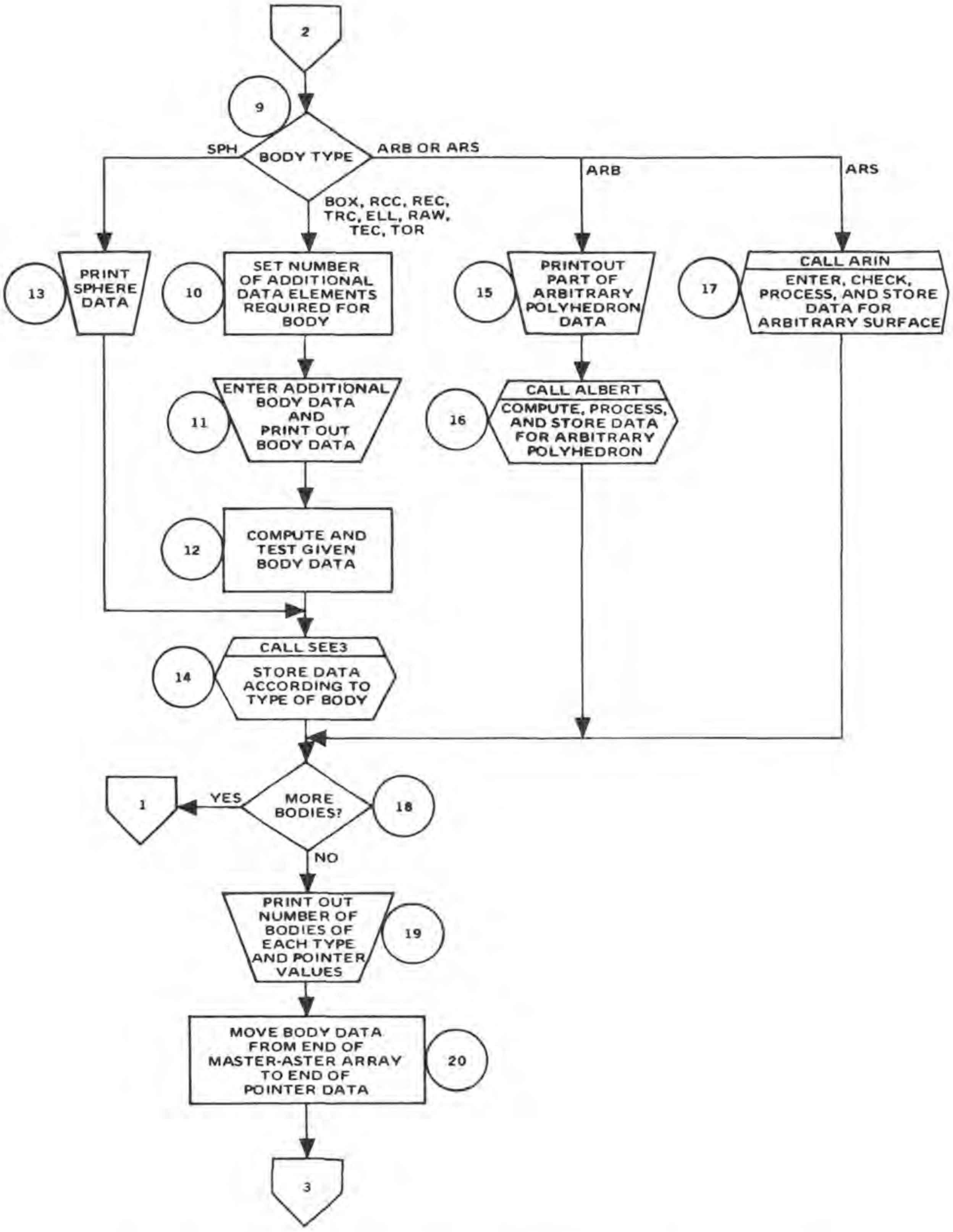

2
9
BODY TYPE
SPH
ARB OR ARS
ARB
ARS
BOX, RCC, REC, TRC, ELL, RAW, TEC, TOR
13
PRINT SPHERE DATA
10
SET NUMBER OF ADDITIONAL DATA ELEMENTS REQUIRED FOR BODY
15
PRINTOUT PART OF ARBITRARY POLYHEDRON DATA
17
CALL ARIN
ENTER, CHECK, PROCESS, AND STORE DATA FOR ARBITRARY SURFACE
11
ENTER ADDITIONAL BODY DATA AND PRINT OUT BODY DATA
16
CALL ALBERT
COMPUTE, PROCESS, AND STORE DATA FOR ARBITRARY POLYHEDRON
12
COMPUTE AND TEST GIVEN BODY DATA
14
CALL SEE3
STORE DATA ACCORDING TO TYPE OF BODY
1
YES
MORE BODIES?
18
NO
PRINT OUT NUMBER OF BODIES OF EACH TYPE AND POINTER VALUES
19
MOVE BODY DATA FROM END OF MASTER-ASTER ARRAY TO END OF POINTER DATA
20
3

FIG. 50. Subroutine GENI Concept Flowchart (Continued)

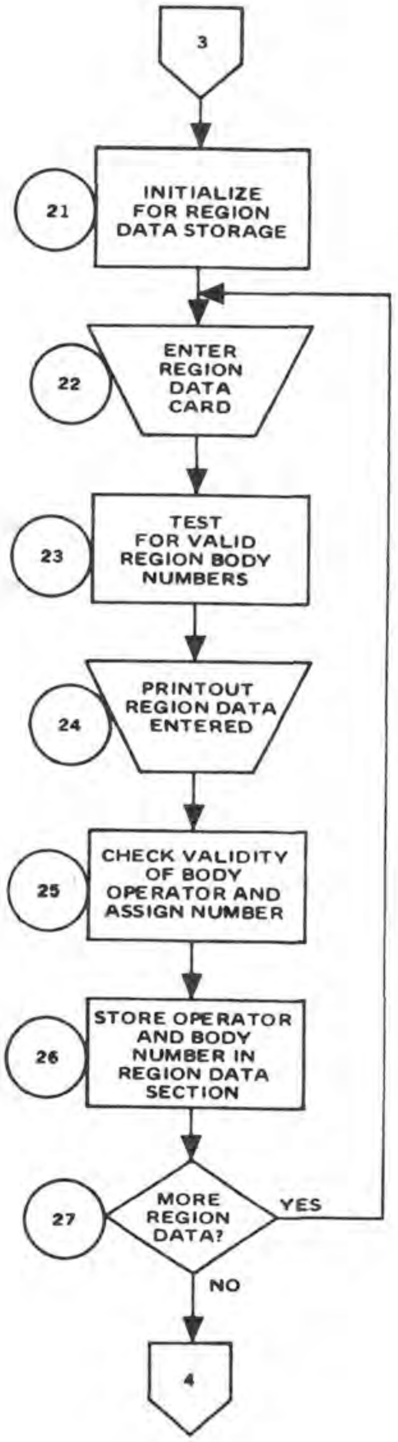

3
21
INITIALIZE FOR REGION DATA STORAGE
22
ENTER REGION DATA CARD
23
TEST FOR VALID REGION BODY NUMBERS
24
PRINTOUT REGION DATA ENTERED
25
CHECK VALIDITY OF BODY OPERATOR AND ASSIGN NUMBER
26
STORE OPERATOR AND BODY NUMBER IN REGION DATA SECTION
27
MORE REGION DATA?
YES
NO
4

FIG. 50. Subroutine GENI Concept Flowchart (Continued)

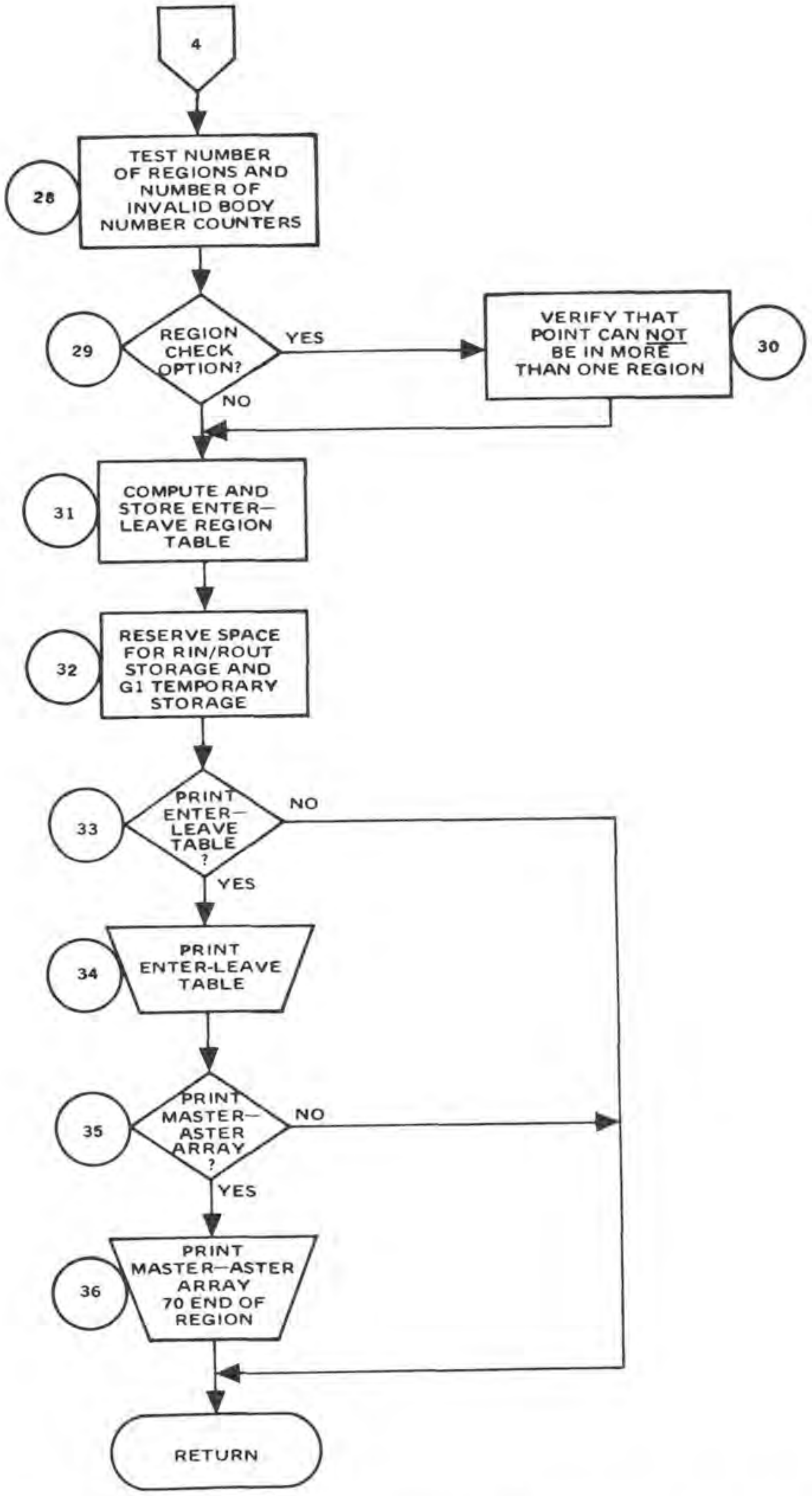

4
28
TEST NUMBER OF REGIONS AND NUMBER OF INVALID BODY NUMBER COUNTERS
29
REGION CHECK OPTION?
YES
NO
VERIFY THAT POINT CAN NOT BE IN MORE THAN ONE REGION
30
31
COMPUTE AND STORE ENTER—LEAVE REGION TABLE
32
RESERVE SPACE FOR RIN/ROUT STORAGE AND G1 TEMPORARY STORAGE
33
PRINT ENTER—LEAVE TABLE ?
NO
YES
34
PRINT ENTER-LEAVE TABLE
35
PRINT MASTER—ASTER ARRAY ?
NO
YES
36
PRINT MASTER—ASTER ARRAY 70 END OF REGION
RETURN

FIG. 50. Subroutine GENI Concept Flowchart (Concluded)

INPUT OF RECTANGULAR PARALLELEPIPED (SUBROUTINE RPPIN)

Subroutine RPPIN is called by Subroutine GENI during the geometry input and processing phase of the MAGIC program if rectangular parallelepiped (RPP) data is to be entered. This subroutine enters the RPP data, computes the number of abutting RPP's to each side, computes the location of those abutting RPP's, computes the location of the boundary coordinate for each side of each RPP in the target geometry, and eliminates redundant boundary coordinates in the RPP coordinate data. The validity of the RPP data is checked if more than one RPP is used to describe the target geometry.

The following description is an outline of the purpose and flow of Subroutine RPPIN. The steps that describe Subroutine RPPIN follow the steps of the conceptual flowchart of Figure 51.

Step 1

Initialize counters and storage pointers for the starting location of the pointer data and the beginning location of the RPP boundary coordinates.

Step 2

Enter and print out the six boundary coordinates for the current RPP.

Step 3

Test the x, y, or z pair of boundary coordinates of the given RPP to verify that the maximum boundary coordinate is greater than the minimum boundary coordinate. If this is not true, print out an error message and stop the program.

Step 4

Test to determine if all coordinate pairs of the given RPP have been tested. If not, branch to test the next pair.

Step 5

Store boundary coordinates of RPP and pointer to boundary coordinates. Check for redundant boundary coordinates such as for abutting RPP's, and eliminate to save storage space. In such a case, only the pointer to the redundant coordinate is stored.

Step 6

Test to determine if more RPP data is to be entered. If not, compute the pointer for the beginning of the abutting RPP data and set pointers to the last location of the boundary coordinate data.

Step 7

Begin loop for checking each side of each RPP for abutting RPP surfaces.

Step 8

Test each side of each RPP for abutting RPP's and store the count. For a detailed discussion of determining the abutting RPP's and other calculations in this subroutine, refer to the discussion in the simulation model.

Step 9

Test to determine if more RPP's are to be checked for abutting RPP's.

Step 10

Test validity of RPP data, if there is more than one RPP in the target geometry, to verify that for each side of an RPP there is either an abutting RPP, or if the side faces the outside of the geometry that it has the same boundary coordinate as those same sides of the other RPP's in the geometry whose sides are on the same outside boundary. Set variable to last word of RPP data and return control to Subroutine GENI.

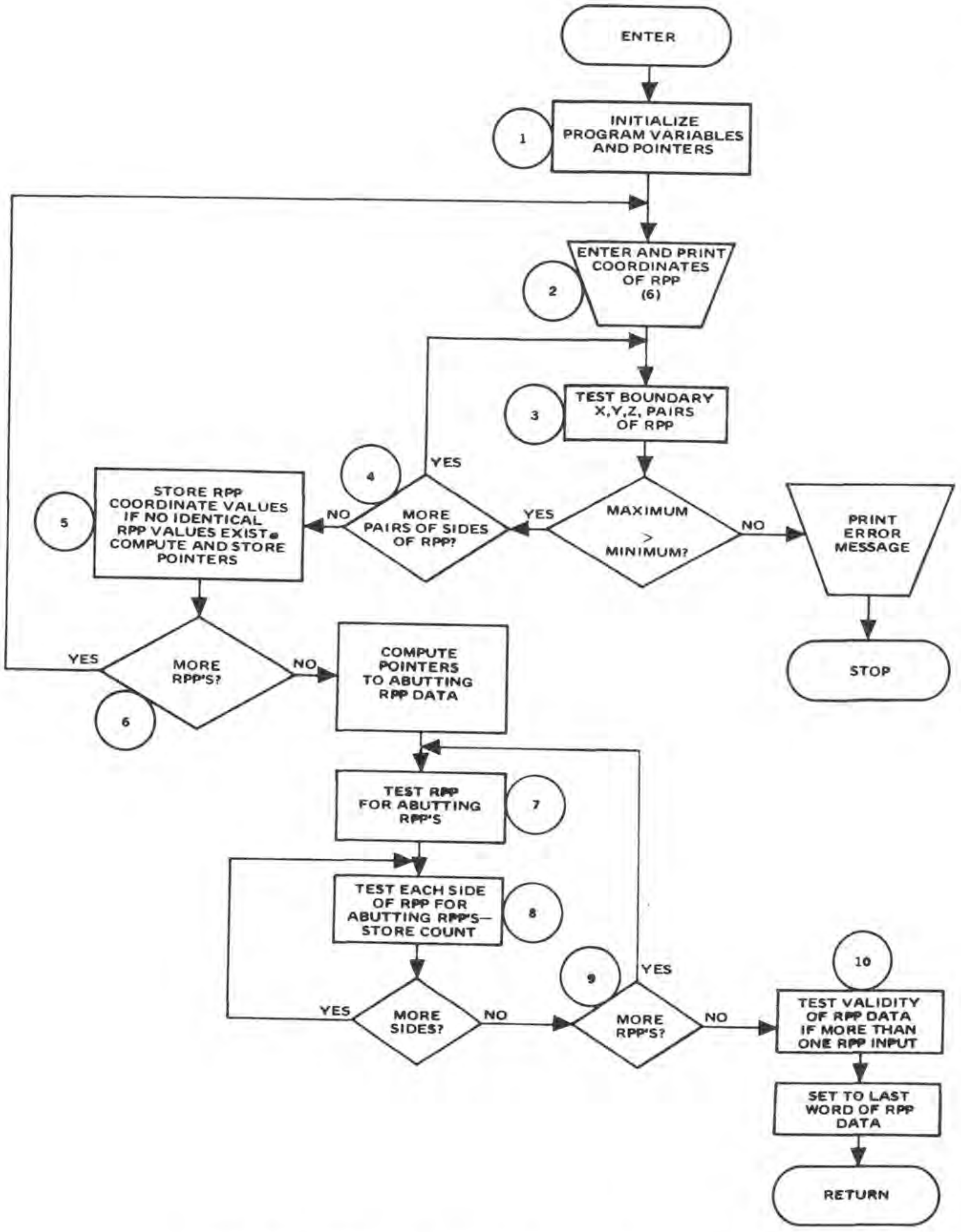

ENTER
1
INITIALIZE PROGRAM VARIABLES AND POINTERS
2
ENTER AND PRINT COORDINATES OF RPP (6)
3
TEST BOUNDARY X,Y,Z, PAIRS OF RPP
MAXIMUM > MINIMUM?
YES
NO
PRINT ERROR MESSAGE
STOP
4
MORE PAIRS OF SIDES OF RPP?
YES
NO
5
STORE RPP COORDINATE VALUES IF NO IDENTICAL RPP VALUES EXIST. COMPUTE AND STORE POINTERS
6
MORE RPP'S?
YES
NO
COMPUTE POINTERS TO ABUTTING RPP DATA
7
TEST RPP FOR ABUTTING RPP'S
8
TEST EACH SIDE OF RPP FOR ABUTTING RPP'S— STORE COUNT
MORE SIDES?
YES
NO
9
MORE RPP'S?
YES
NO
10
TEST VALIDITY OF RPP DATA IF MORE THAN ONE RPP INPUT
SET TO LAST WORD OF RPP DATA
RETURN

FIG. 51. Subroutine RPPIN Concept Flowchart

INPUT OF ARBITRARY POLYHEDRON (SUBROUTINE ALBERT)

Subroutine ALBERT is called by Subroutine GENI during the geometry input and processing phase of the MAGIC program whenever input data for an arbitrary polyhedron is to be entered. Subroutine ALBERT computes the equation of each side of six possible sides of the ARB, verifies that all four vertices of a given plane are in the same plane, determines the relative position of the side with respect to the other sides of the ARB, and stores the coefficients of the plane equation in the MASTER-ASTER array along with the location of the data for each plane.

The following description is an outline of the purpose and flow of Subroutine ALBERT. The steps that describe Subroutine ALBERT follow the steps of the conceptual flowchart of Figure 52.

Step 1

The coordinates of the first two vertices of the ARB are entered on the body data card by Subroutine GENI and passed to this subroutine. These coordinates are stored in an array used for storing all of the coordinates of the vertices of the ARB.

Step 2

Enter the coordinates of the remaining six vertices and the four ordinal vertex numbers for each of the six faces. The coordinates of the eight vertices of the ARB and the ordinal vertex numbers for the six sides are then printed out.

Step 3

Begin loop to compute the coefficients for the equation of the six planes of the ARB, vertify that all of the points of a side lie in a plane, and determine the relative position of the side with respect to the other sides of the ARB. This step retrieves the first three vertex coordinates of the given plane and computes the coefficients A, B, C, and D of the equation of a plane

$$Ax + By + Cz + D = 0$$

using the three-point form

$$\begin{vmatrix} y_1 & z_1 & 1 \\ y_2 & z_2 & 1 \\ y_3 & z_3 & 1 \end{vmatrix} x + \begin{vmatrix} z_1 & x_1 & 1 \\ z_2 & x_2 & 1 \\ z_3 & x_3 & 1 \end{vmatrix} y + \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} z - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} = 0$$

Therefore

$$A = -y_2z_3+z_2y_3+y_1z_3-z_1y_3-y_1z_2+z_1y_2$$

$$B = x_2z_3-z_2x_3-x_1z_3+x_3z_1+x_1z_2-x_2z_1$$

$$C = y_2x_3-y_3x_2-y_1x_3+y_3x_1+y_1x_2-y_2x_1$$

$$D = x_1(y_2z_3-y_3z_2) - x_2(y_1z_3-y_3z_1) + x_3(y_1z_2-y_2z_1)$$

Step 4

Test for a degenerate plane by testing the sum of the square of the coefficients

$$A^2 + B^2 + C^2$$

for a value of zero. If zero, the message DEGENERATE PLANE is printed out, the constant term is set to its absolute value and the program branches to store the data for the plane.

Step 5

Test for an undefined plane by testing the sum of the square of the coefficients against a very small value of the square of the length between two of the vertices of the plane. If the sum of the squares is not greater, the message UNDEFINED PLANE along with data about the plane is printed out and the program branches to consider the next plane.

Step 6

Compute the direction cosines of a perpendicular from the origin to the plane being considered where these direction cosines are defined as

$$\cos \alpha_x = A\Big/\sqrt{A^2 + B^2 + C^2}$$

$$\cos \alpha_y = B\Big/\sqrt{A^2 + B^2 + C^2}$$

$$\cos \alpha_z = C\Big/\sqrt{A^2 + B^2 + C^2}$$

Step 7

Retrieve coordinates of the fourth vertex of plane and compute perpendicular distance from point to plane formed by other three vertices using the distance from a point (x_1, y_1, z_1) to a plane $(Ax + By + Cz + D = 0)$ relationship

$$\text{distance} = \frac{Ax_1 + By_1 + Cz_1 + D}{\sqrt{A^2 + B^2 + C^2}}$$

Step 8

Test to determine if the fourth vertex lies on the plane by comparing the square of the distance with a given tolerance. If not within tolerance (not on plane), print out FOUR POINTS NOT IN PLANE along with data about the plane and branch to consider the next plane.

Step 9

Retrieve the four other vertices of the ARB and substitute each in the plane equation for the current plane and store the four results for tests.

Step 10

Test results of Step 9 to determine relative position of the remaining four points with respect to current plane (in back of, on, or in front of).

Step 11

Test to determine if remaining four points are all on the same side of the current plane. If not, there is an error in the ARB data, so the message ERROR IN SIDE DESCRIPTION along with data about the current plane and the remaining four vertices are printed out.

Step 12

Reverse coefficients of current plane of the remaining four vertices to establish the sign convention of the ARB such that the positive side of the given plane is on the interior of the ARB. This insures that the normal for each plane is directed into the ARB.

Step 13

Call Subroutine SEE3 to store the coefficients and the constant term in the ASTER array, and to compute the pointer to the data and store in the MASTER array.

Step 14

Test to determine if all of the sides of the ARB have been computed, tested, and stored; if not, the program branches to consider the next side.

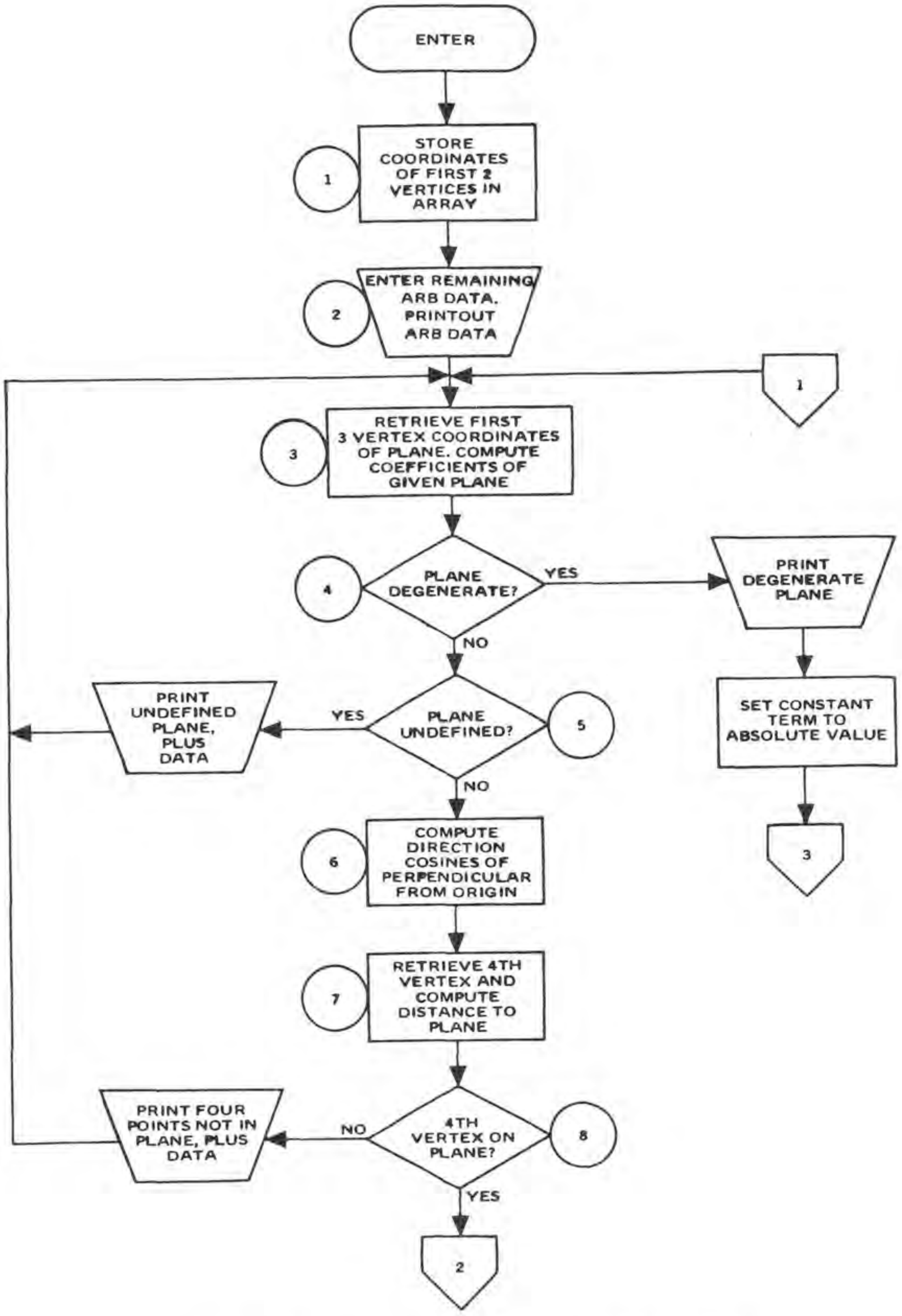

ENTER
1
STORE COORDINATES OF FIRST 2 VERTICES IN ARRAY
2
ENTER REMAINING ARB DATA. PRINTOUT ARB DATA
1
3
RETRIEVE FIRST 3 VERTEX COORDINATES OF PLANE. COMPUTE COEFFICIENTS OF GIVEN PLANE
4
PLANE DEGENERATE?
YES
PRINT DEGENERATE PLANE
NO
SET CONSTANT TERM TO ABSOLUTE VALUE
3
5
PLANE UNDEFINED?
YES
PRINT UNDEFINED PLANE, PLUS DATA
NO
6
COMPUTE DIRECTION COSINES OF PERPENDICULAR FROM ORIGIN
7
RETRIEVE 4TH VERTEX AND COMPUTE DISTANCE TO PLANE
8
4TH VERTEX ON PLANE?
NO
PRINT FOUR POINTS NOT IN PLANE, PLUS DATA
YES
2

FIG. 52. Subroutine ALBERT Concept Flowchart

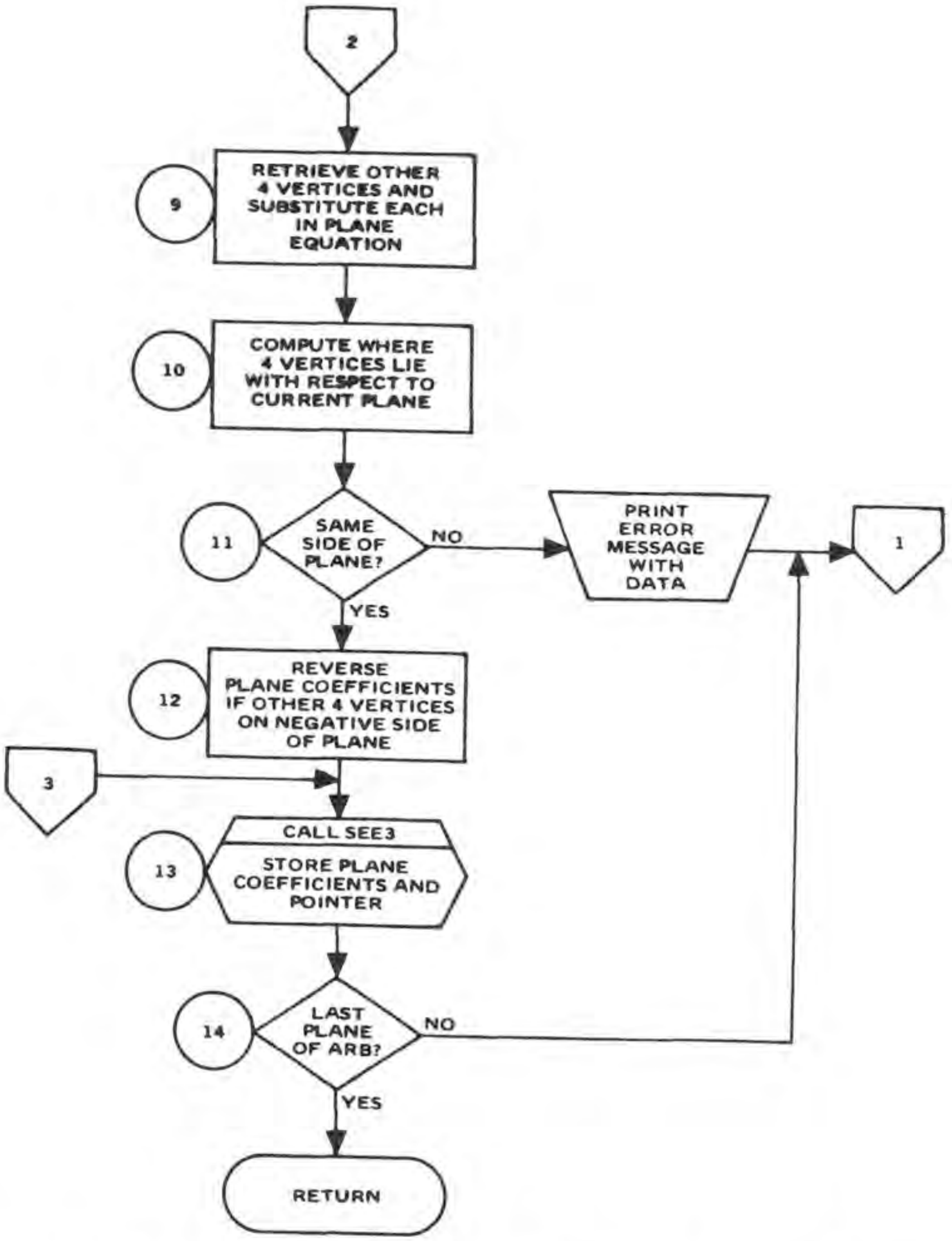

2
9
RETRIEVE OTHER 4 VERTICES AND SUBSTITUTE EACH IN PLANE EQUATION
10
COMPUTE WHERE 4 VERTICES LIE WITH RESPECT TO CURRENT PLANE
11
SAME SIDE OF PLANE?
NO
PRINT ERROR MESSAGE WITH DATA
1
YES
12
REVERSE PLANE COEFFICIENTS IF OTHER 4 VERTICES ON NEGATIVE SIDE OF PLANE
3
CALL SEE3
13
STORE PLANE COEFFICIENTS AND POINTER
14
LAST PLANE OF ARB?
NO
YES
RETURN

FIG. 52. Subroutine ALBERT Concept Flowchart (Concluded)

INPUT OF ARBITRARY SURFACE (SUBROUTINE ARIN)

Subroutine ARIN is called by Subroutine GENI during the input processing phase of the MAGIC program to enter, check, process, and store the data elements of a given arbitrary surface (ARS) in the MASTER-ASTER array.

The following description is an outline of the purpose and flow of Subroutine ARIN. The steps that describe Subroutine ARIN follow the steps of the conceptual flowchart of Figure 53.

Step 1

Enter the number of curves (M) used to describe the ARS and the number of points on each curve (N).

Step 2

Compute the total number of points used to describe the ARS (number of curves times the number of points/curve, M*N), the number of points to be stored (NP=2N(M-1)), and the number of storage words required for the given ARS. The information is then printed out.

Step 3

Test to determine if there were curve and point data entered for the ARS. If not, an error message is printed out and the program returns control to Subroutine GENI.

Step 4

Reserve storage in the body data section of the ASTER array for the ARS data, and set and store points for the body data.

Step 5

Begin loop for each curve of the ARS to enter the coordinates of each point and store them in the NP=2N(M-1) storage locations. NP locations are required since each point requires four storage words for the three coordinates and a flag to specify whether the point along with the following two points form a valid (0) or invalid (-1) triangle. Also, the points are stored in pairs between curves. For example, given an ARS of five curves (M=5) with four points per curve (N=4), the total points to be entered would be $M \cdot N=5 \cdot 4=20$, but would require NP=2N(M-1) $=2 \cdot 4(5-1)=32$ points of storage (four words per point). These points, P_{MN}, would be stored in the following sequence: P_{11},P_{21}; P_{12},P_{22}; P_{13},P_{23}; P_{14},P_{24}; P_{21},P_{31}; $\cdots$ P_{43},P_{53}; P_{44},P_{54}.

Step 6

Test to determine if the points for more curves on the ARS are to be entered. If not, the number of points stored, NP=2N(M-1), is stored in the first word of ARS data and the second word is initialized to zero for use as a hit counter. Compute and print out the maximum number of possible triangles on the ARS.

Step 7

Test each three consecutive points to determine if they form a triangle by computing the cross product of two legs of the triangle and the dot product of the result. If zero, the three points form a degenerate triangle and a -1 is stored in the fourth word of the first of the three points. The number of valid triangles would then be reduced by one.

Step 8

Print out the number of valid triangles on the surface of the ARS and return control to Subroutine GENI.

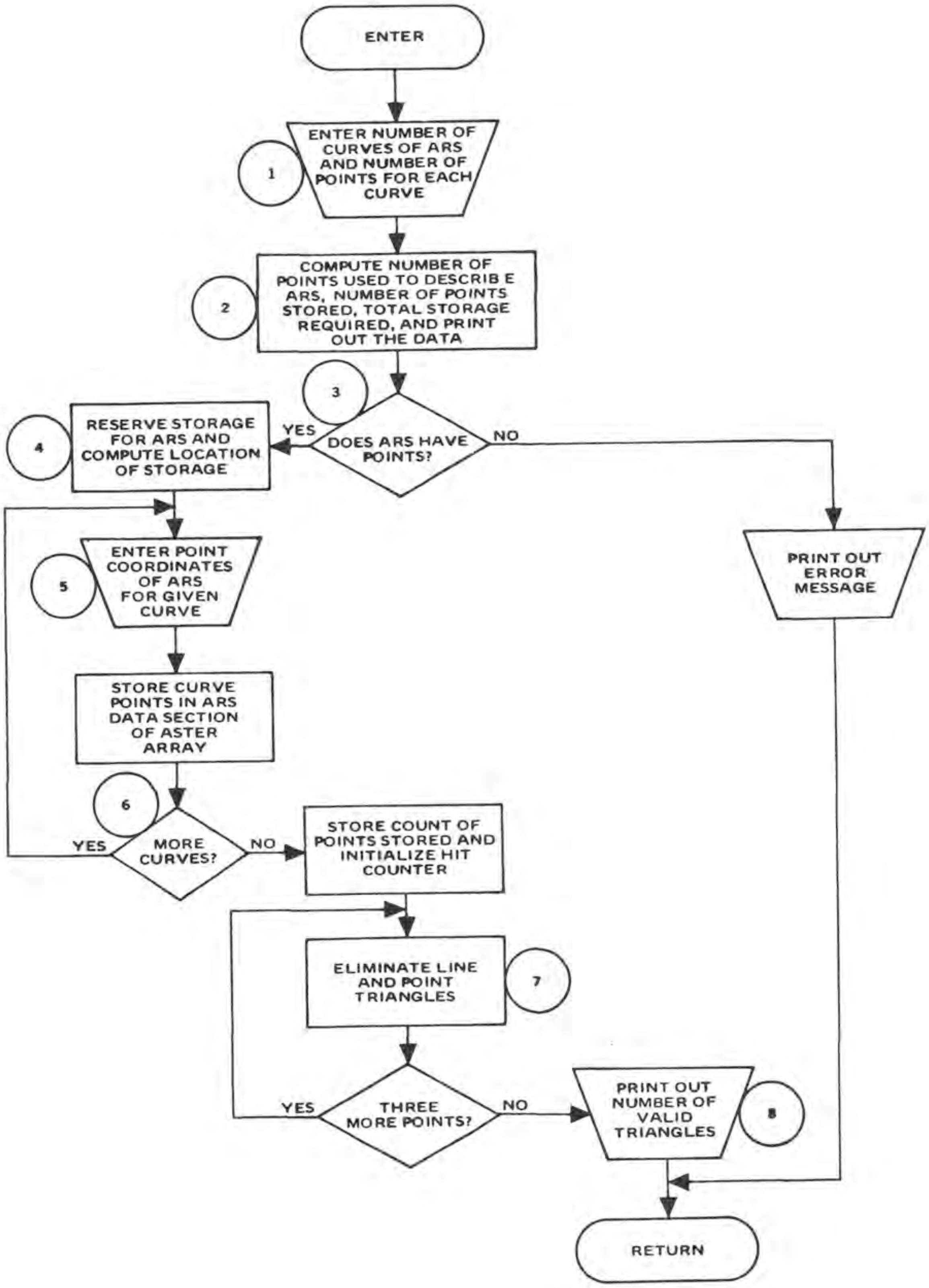

ENTER
1
ENTER NUMBER OF CURVES OF ARS AND NUMBER OF POINTS FOR EACH CURVE
2
COMPUTE NUMBER OF POINTS USED TO DESCRIB E ARS, NUMBER OF POINTS STORED, TOTAL STORAGE REQUIRED, AND PRINT OUT THE DATA
3
DOES ARS HAVE POINTS?
YES
NO
4
RESERVE STORAGE FOR ARS AND COMPUTE LOCATION OF STORAGE
PRINT OUT ERROR MESSAGE
5
ENTER POINT COORDINATES OF ARS FOR GIVEN CURVE
STORE CURVE POINTS IN ARS DATA SECTION OF ASTER ARRAY
6
MORE CURVES?
YES
NO
STORE COUNT OF POINTS STORED AND INITIALIZE HIT COUNTER
ELIMINATE LINE AND POINT TRIANGLES
7
THREE MORE POINTS?
YES
NO
PRINT OUT NUMBER OF VALID TRIANGLES
8
RETURN

FIG. 53. Subroutine ARIN Concept Flowchart

GRID/ATTACK PLANE (SUBROUTINE GRID)

Subroutine GRID is the main control program for tracking a ray through the target geometry. It is called by the main control program of MAGIC and controls all of the input and processing for a single attack plane after reading in data and generating the attack plane at the given angle of attack. This plane is divided into a grid of square cells, with each cell acting as the point of origin of a ray (an option is available to skip a random number of cells). The tracing of each ray from the grid through the target geometry is accomplished by Subroutine TRACK. Subroutine GRID therefore calls Subroutine TRACK once for each ray or shotline after Subroutine GRID has defined the ray.

The following description is an outline of the purpose and flow of the MAIN program. The steps that describe the program follow the steps of the conceptual flowchart of Figure 54.

Step 1

Enter the grid input parameters, initialize specific parameters if not set by the input, and print the grid input parameters.

Step 2

Compute the sine and cosine of the attack azimuth and elevation angles.

Step 3

Compute the direction cosines of the rays perpendicular to the grid plane and directed toward the target geometry.

Step 4

Compute the row and column number of the current grid cell from which a ray is to be fired through the target geometry.

Step 5

Compute the coordinates of the current grid cell with respect to the center of the grid plane.

Step 6

Test to determine if ray or shotline is to be fired from the center of the grid cell from a random point within the cell and compute the coordinates of the origin accordingly.

Step 7

Transform the origin of the shotline coordinates into the coordinate system of the target and shift the origin if the input specified a shift.

Step 8

Call Subroutine TROPIC to generate random direction cosines. The ray origin is then moved by a very small amount in a random direction.

Step 9

Back out the origin of the ray from the grid plane within the target to an effective attack plane from where the ray will be fired toward the target.

Step 10

If the origin of the ray is now below the defined ground plane, the ray is not fired for the current cell. If above ground, the starting coordinates and direction of the ray are saved for later reference.

Step 11

Call Subroutine TRACK to coordinate all the processing for the ray until it emerges from the far side of the vehicle, and to provide the calculated output for the ray. Control will not return to Subroutine GRID until calculations and outputs of results are complete for the given grid cell.

Step 12

Test to determine if all grid plane cells are to be processed or if a random number is to be skipped after each cell is processed.

Step 13

Increment to the next sequential cell to generate and process more cell data. If all cells have been processed, return control to the MAIN program.

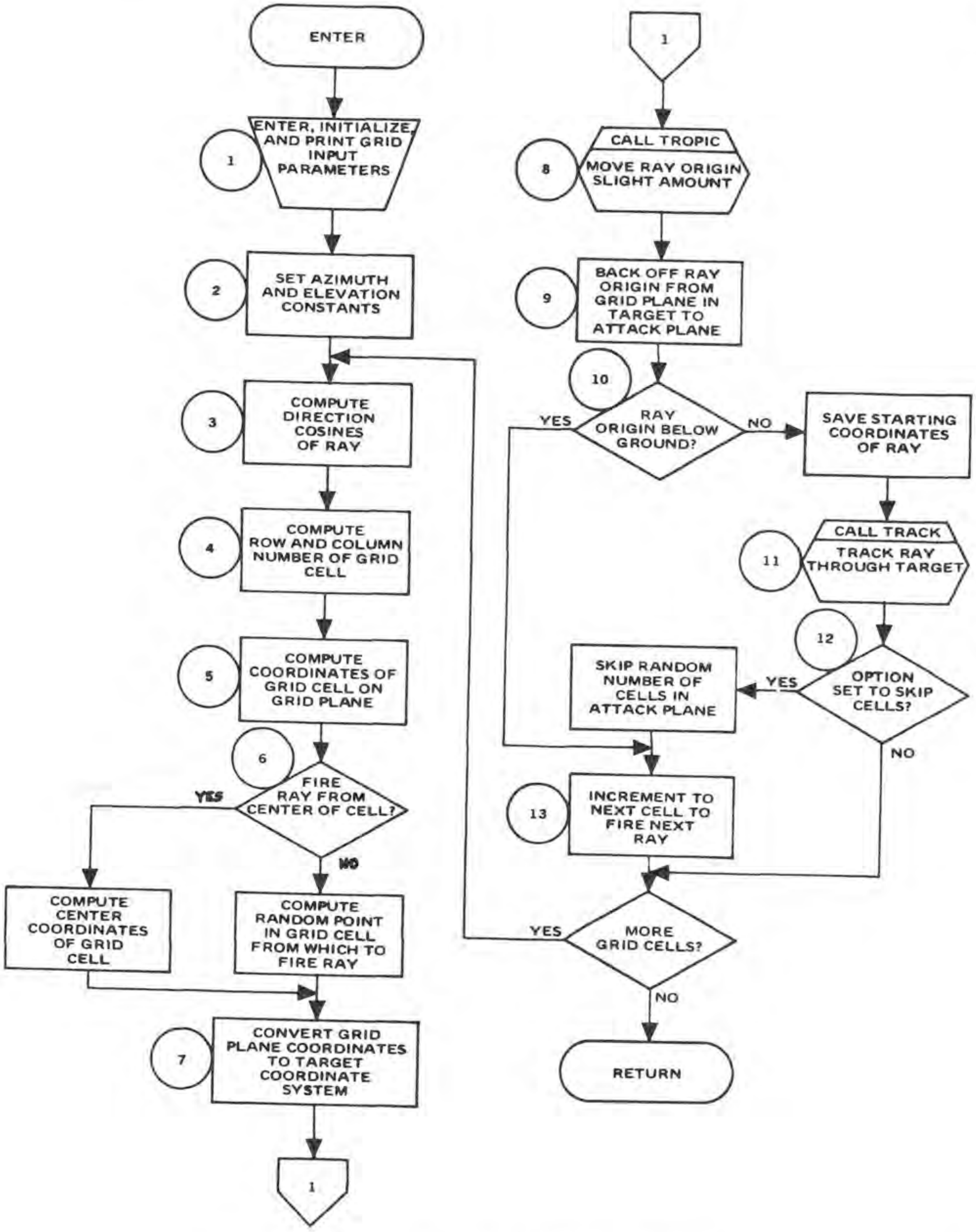

ENTER
1
ENTER, INITIALIZE, AND PRINT GRID INPUT PARAMETERS
2
SET AZIMUTH AND ELEVATION CONSTANTS
3
COMPUTE DIRECTION COSINES OF RAY
4
COMPUTE ROW AND COLUMN NUMBER OF GRID CELL
5
COMPUTE COORDINATES OF GRID CELL ON GRID PLANE
6
FIRE RAY FROM CENTER OF CELL?
YES
NO
COMPUTE CENTER COORDINATES OF GRID CELL
COMPUTE RANDOM POINT IN GRID CELL FROM WHICH TO FIRE RAY
7
CONVERT GRID PLANE COORDINATES TO TARGET COORDINATE SYSTEM
1
1
8
CALL TROPIC
MOVE RAY ORIGIN SLIGHT AMOUNT
9
BACK OFF RAY ORIGIN FROM GRID PLANE IN TARGET TO ATTACK PLANE
10
RAY ORIGIN BELOW GROUND?
YES
NO
SAVE STARTING COORDINATES OF RAY
11
CALL TRACK
TRACK RAY THROUGH TARGET
12
OPTION SET TO SKIP CELLS?
YES
NO
SKIP RANDOM NUMBER OF CELLS IN ATTACK PLANE
13
INCREMENT TO NEXT CELL TO FIRE NEXT RAY
MORE GRID CELLS?
YES
NO
RETURN

FIG. 54. Subroutine GRID Concept Flowchart

RAY TRACKING (SUBROUTINE TRACK)

Subroutine TRACK has the function of accepting grid coordinates from Subroutine GRID and initializing the "firing" of a ray. It follows the ray through all points of contact and outputs cell identification data and the intersect data for each ray for subsequent use in vulnerability analysis studies.

The following description is an outline of the purpose and flow of Subroutine TRACK. The following steps that describe Subroutine TRACK follow the steps of the conceptual flowchart of Figure 55.

Step 1

Initialize subroutine constants, the starting region of the ray, the number of intersections, the number of components hit, the number of spaces counter, and the target, interior volume, armor, and skirt flags. Also initialize to zero the surface number-body number-next region number array and the line-of-sight distance from contact to contact array.

Step 2

Call Subroutine G1 to find the distance to the next region, the number of the next region, and the new position of the point along the ray.

Step 3

If the region number returned by Subroutine G1 is not negative (an error if negative and control is returned to Subroutine GRID), the line-of-sight distance to the new point and the surface number, the body number, and the next region number are stored.

Step 4

Test to determine if the new intersect occurs at an RPP boundary, which means that the end of the ray has been reached. If the end of the ray, branch to output the intersect data of the ray.

Step 5

Test to determine if the present intersect is the first along the ray if the end of the ray has not been reached. If not the first intersect, branch around the computations involved for the first intersect.

Step 6

If the first intersect along the ray, compute the distance along the ray path from the first intersect to the original point on the grid plane within the target.

Step 7

If the intersect is not the first along the ray, test for special material in the region just traversed and update the hit counter and set the space flag and ident flags accordingly. If the maximum of the storage arrays have not yet been reached, branch to compute the data for the next intersect.

Step 8

When the end of the ray has been reached and there have been intersects along the ray, the distance from the final intersect to the original point on the grid plane within the target is computed.

Step 9

Output cell identification data for the first line of output for the given ray. The cell ID data consists of the cell number reference, the distances to the center grid plane from the initial and final intersects, the distance to the ray point from the grid plane counter, the number of intersects, and the space and ident flags.

Step 10

Call Subroutine CALC to compute cell intersection data for the first half of line of output. Subroutine CALC will compute, for each intersect, the region identification, line-of-sight distance, surface intersect obliquity angle, normal distance through region, type of space following present region, and the line-of-sight distance through the space.

Step 11

Test to determine if the data for the last intersect has been computed. If it has, set variables to zero for second half of line of output and branch around the computation of data for the second half line of output.

Step 12

Call Subroutine CALC to compute information on the next intersect for the second half of the present line of data. Data returned by Subroutine CALC is identical to that of Step 10.

Step 13

Update the number of spaces hit and the number of components hit counters, and output the line of ray intersect data (two components).

Step 14

Test to determine if more ray intersect data is to be computed and output. If there is, branch to process data for the next line of output.

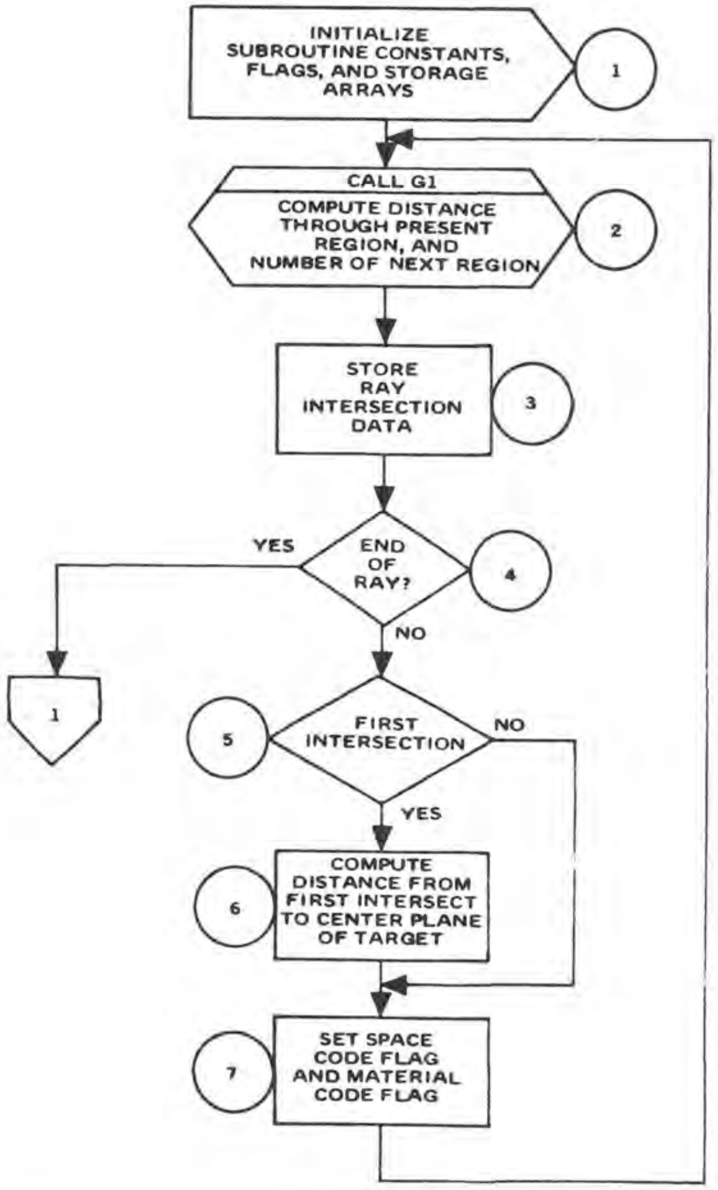

INITIALIZE SUBROUTINE CONSTANTS, FLAGS, AND STORAGE ARRAYS
1
CALL G1
COMPUTE DISTANCE THROUGH PRESENT REGION, AND NUMBER OF NEXT REGION
2
STORE RAY INTERSECTION DATA
3
END OF RAY?
4
YES
NO
1
FIRST INTERSECTION
5
NO
YES
COMPUTE DISTANCE FROM FIRST INTERSECT TO CENTER PLANE OF TARGET
6
SET SPACE CODE FLAG AND MATERIAL CODE FLAG
7

FIG. 55. Subroutine TRACK Concept Flowchart

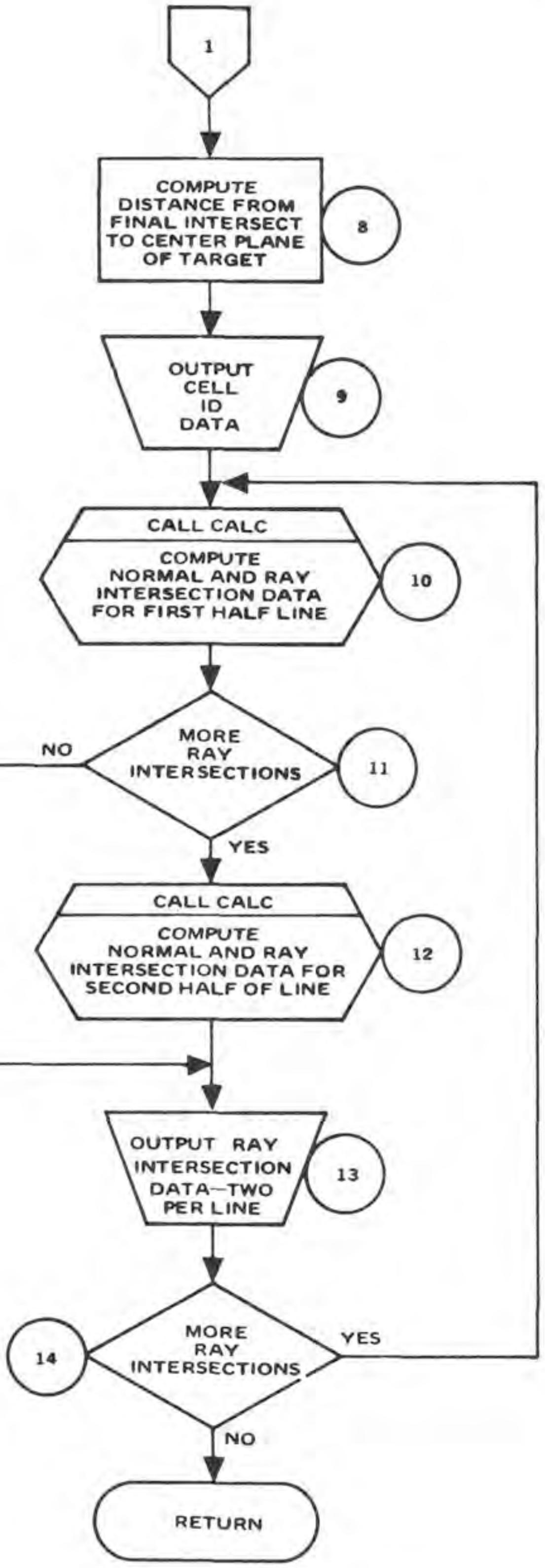

1
COMPUTE DISTANCE FROM FINAL INTERSECT TO CENTER PLANE OF TARGET
8
OUTPUT CELL ID DATA
9
CALL CALC
COMPUTE NORMAL AND RAY INTERSECTION DATA FOR FIRST HALF LINE
10
MORE RAY INTERSECTIONS
11
NO
YES
CALL CALC
COMPUTE NORMAL AND RAY INTERSECTION DATA FOR SECOND HALF OF LINE
12
OUTPUT RAY INTERSECTION DATA—TWO PER LINE
13
MORE RAY INTERSECTIONS
14
YES
NO
RETURN

FIG. 55. Subroutine TRACK Concept Flcwchart (Concluded)

NORMAL AND ANGLE OF OBLIQUITY CALCULATIONS (SUBROUTINE CALC)

Subroutine CALC is called by Subroutine TRACK to compute the normal distance and the angle of obliquity for each region that the ray passes through. Subroutine TRACK calls Subroutine CALC to perform these calculations after the ray has passed through all of the target geometry. Subroutine CALC also assigns a 9, indicating termination of the ray, to the space following the RPP containing the target geometry.

The following description is an outline of the purpose and flow of the MAIN program. The steps that describe the program follow the steps of the conceptual flowchart of Figure 56.

Step 1

Retrieve the surface number, body number, and the following region for the present intersect from the array of intersect data compiled by Subroutine TRACK.

Step 2

Retrieve the line-of-sight distance through the region following the intersect and update the contact point coordinates to the present intersect position. The travel distance of the ray is updated by adding the distance through the region following the present intersect.

Step 3

Retrieve the body type and the location of other descriptive body data for the body where the present intersect occurs.

Step 4

Test the validity of the body type and branch, according to the body type of the intersect being considered, to find the direction cosines of the normal at the intersect (into the body for an entry intersect and away from the body for an exit intersect).

Step 5

Compute the angle between the normal at the intersect and the direction of the ray.

Step 6

Call Subroutine G1 to compute the normal distance through the region following the intersect.

Step 7

Retrieve the space code of the region following the next intersect and the item or component code of the region following the present intersect.

Step 8

Test to determine if the region following the next intersect has a special identification (other than space). If so, return control to Subroutine TRACK.

Step 9

Test to determine if the end of the ray has been reached by testing the next intersect to determine if it is an intersect with the enclosing RPP. If it is, flags are set to indicate the end of the ray and the program returns control to Subroutine TRACK.

Step 10

Retrieve the distance through the space of the region following the present intersect and update the line-of-sight distance traveled by the ray before returning to Subroutine TRACK, if the next intersect is not an RPP intersect.

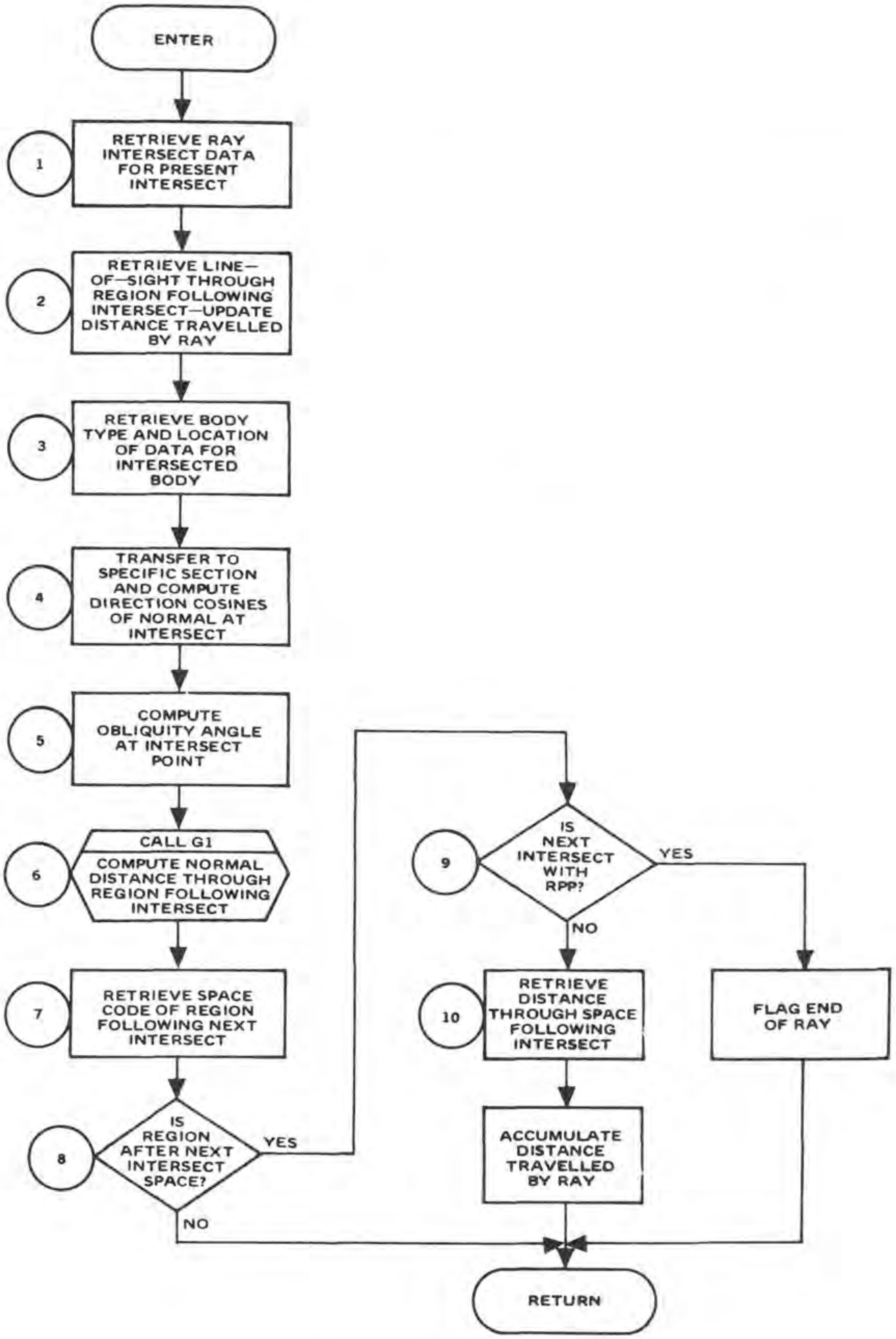

ENTER
1
RETRIEVE RAY INTERSECT DATA FOR PRESENT INTERSECT
2
RETRIEVE LINE—OF—SIGHT THROUGH REGION FOLLOWING INTERSECT—UPDATE DISTANCE TRAVELLED BY RAY
3
RETRIEVE BODY TYPE AND LOCATION OF DATA FOR INTERSECTED BODY
4
TRANSFER TO SPECIFIC SECTION AND COMPUTE DIRECTION COSINES OF NORMAL AT INTERSECT
5
COMPUTE OBLIQUITY ANGLE AT INTERSECT POINT
6
CALL G1
COMPUTE NORMAL DISTANCE THROUGH REGION FOLLOWING INTERSECT
7
RETRIEVE SPACE CODE OF REGION FOLLOWING NEXT INTERSECT
8
IS REGION AFTER NEXT INTERSECT SPACE?
YES
NO
9
IS NEXT INTERSECT WITH RPP?
YES
NO
10
RETRIEVE DISTANCE THROUGH SPACE FOLLOWING INTERSECT
ACCUMULATE DISTANCE TRAVELLED BY RAY
FLAG END OF RAY
RETURN

FIG. 56. Subroutine CALC Concept Flowchart

RAY INTERSECTIONS (SUBROUTINE G1)

Subroutine G1 is the main ray-tracing routine of the MAGIC program. It performs the following function: Given a ray in a region at a specified location with direction cosines, find the distance to the next region and the number of the next region.

The following description is an outline of the purpose and flow of Subroutine G1. The following steps that describe Subroutine G1 follow the steps of the conceptual flowchart of Figure 57.

Step 1

Initialize the variable that accumulates the total distance to the next region to zero, and initialize a variable indicator for line-of-sight distance or for normal distance through the region.

Step 2

Test to determine if a new ray is to be processed or if the subroutine has been called to continue tracing a ray that has already been started.

Step 3

If a new ray is to be processed, initialize to zero the cumulative distance variable, and initialize the temporary working storage area for Subroutine G1 if it has previously been filled.

Step 4

A point on the ray is stepped through the region by finding the next closest intersect along the ray until a new region is entered. To accomplish this, Subroutine G1 investigates all of the bodies in the region that the ray might strike. Then, for each body in the region description, the distances from the origin to the point where the ray enters the body (RIN) and to the point where the ray leaves the body (ROUT) are computed. Twelve different body routines are available to G1 for this purpose for 12 possible body types that could be in the region. Each intersect distance is compared until the smallest intersect distance is found that is greater than the distance already traveled by the point on the ray. The point on the ray is then advanced to the intersect, recording the body where the intersect occurs.

Step 5

Retrieve from the body data in memory all of the possible regions that the ray could be entering. If the intersect occurs as an entry on the body, the entering region table of that body is retrieved. If the intersect occurs as an exit on the body, the leaving region table of that body is retrieved.

Step 6

Test to determine if there are regions in the region table for the body.

Step 7

Each region from the region table is in turn passed to Subroutine WOWI. Subroutine WOWI tests each region passed to it until the region that the ray is in is located.

Step 8

If there are no regions in the region table, the intersect is located at a leaving surface of a rectangular parallelepiped. Therefore, Subroutine RPP2 is called to identify the abutting rectangular parallelepiped in the path of the ray.

Step 9

Retrieve from the body data in memory all of the possible regions that the ray could be entering for the new RPP.

Step 10

Each region from the entering region table is in turn passed to Subroutine WOWI. Subroutine WOWI tests each region passed to it until the correct region that the ray is in is located.

Step 11

Test the region of the intersect to determine if it is a different region. If a different region, and the subroutine was not called by either Subroutine VOLUM or TESTG, another test is made between the space codes (interior or exterior volume) and between the component codes of the material of the two regions. If the space and material of the two regions are different, the subroutine returns control to Subroutine TRACK. If the space and material of the two regions are identical, then control is returned to the beginning of the subroutine to find the next intersect, etc.

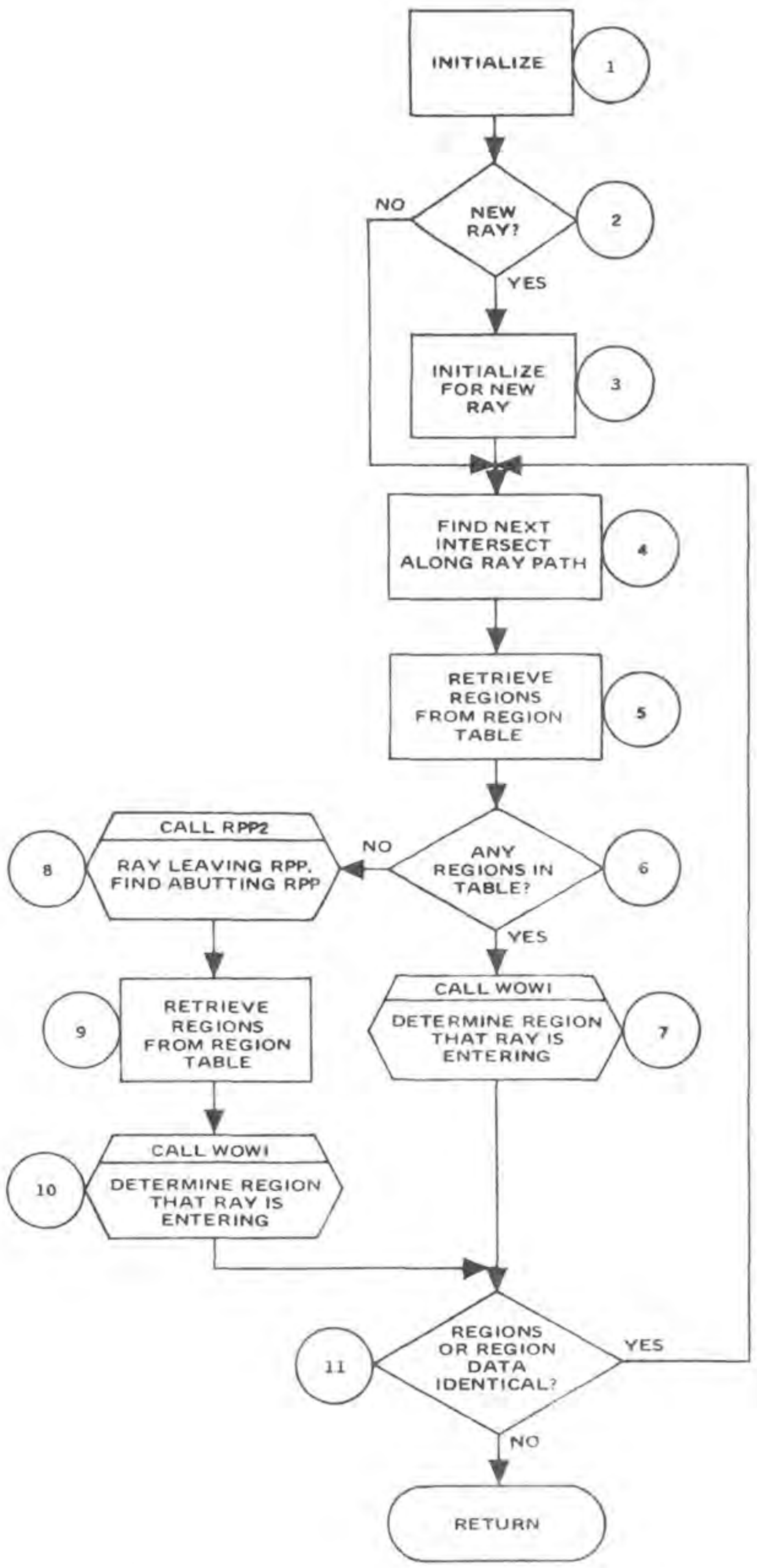

INITIALIZE
1
NEW RAY?
2
NO
YES
INITIALIZE FOR NEW RAY
3
FIND NEXT INTERSECT ALONG RAY PATH
4
RETRIEVE REGIONS FROM REGION TABLE
5
ANY REGIONS IN TABLE?
6
NO
YES
CALL RPP2
RAY LEAVING RPP, FIND ABUTTING RPP
8
CALL WOWI
DETERMINE REGION THAT RAY IS ENTERING
7
RETRIEVE REGIONS FROM REGION TABLE
9
CALL WOWI
DETERMINE REGION THAT RAY IS ENTERING
10
REGIONS OR REGION DATA IDENTICAL?
11
YES
NO
RETURN

FIG. 57. Subroutine GI Concept Flowchart

REGION LOCATION (SUBROUTINE WOWI)

Subroutine WOWI is, in essence, the heart of the combinatorial geometry method. It performs the following function: Given a region number by Subroutine G1, Subroutine WOWI determines if the present position of a specified point along the ray is within that region.

The following description is an outline of the purpose and flow of Subroutine WOWI. The following steps that describe Subroutine WOWI follow the steps of the conceptual flowchart of Figure 58.

Step 1

Retrieve the number of solids that are in the region description. Then retrieve the first solid, its operator, and its body type.

Step 2

Retrieve or calculate the RIN and ROUT intersect distances of the solid measured from the origin of the ray.

Step 3

Test the operator of the solid to determine its relationship within the region where it is located.

Step 4

Test the solid with a (+) operator to determine if the point along the ray is between the RIN and ROUT of the solid, which is required for a valid test of a (+) operator.

Step 5

Test the solid with a (-) operator to determine if the point along the ray misses the solid or is on either side of the solid, which is required for a valid test of a (-) operator.

Step 6

If the test of the solid with either a (+) or (-) operator was valid, then a further test is made to determine if there are more solids in the region, since a region description with no (OR) solid operators is satisfied only if every (+) and (-) operator in the region description is valid.

Step 7

Retrieve the body data for the next solid if there are more solids in region description, and if all (+) and (-) operators of the previous solids in the region description have been valid.

Step 8

Test next solid retrieved for an (OR) operator, since a region description containing one or more (OR) combinations of solids is satisfied if any one of the (OR) combinations is valid.

Step 9

If the region description is satisfied because all (+) and (-) operators of the solids in the region description are valid, or all of the (+) and (-) operators in an (OR) combination of the region description are valid, the point on the ray is within the given region and control is returned to Subroutine G1.

Step 10

If a (+) or (-) operator test was not valid, test to determine if the first operator was an (OR) operator.

Step 11

Test to determine if there are any more solids in the region description. If not, there are no other (OR) combinations in the region description.

Step 12

If there are more solids in the region description, retrieve the next solid and its operator.

Step 13

The ray does not enter the region passed to Subroutine WOWI from Subroutine G2. Therefore, return control to Subroutine G1 to find the next region to be tested by Subroutine WOWI.

Step 14

Test to determine if next solid's operator is an (OR) operator. This test is performed in order to find the next (OR) combination of solids in the region, since a region description containing one or more (OR) combinations is satisfied if any one of the (OR) combinations is valid.

Step 15

If another (OR) combination of solids in the region description is found, then control is transferred to the beginning of the subroutine to determine if this next (OR) combination of bodies in the region description is valid.

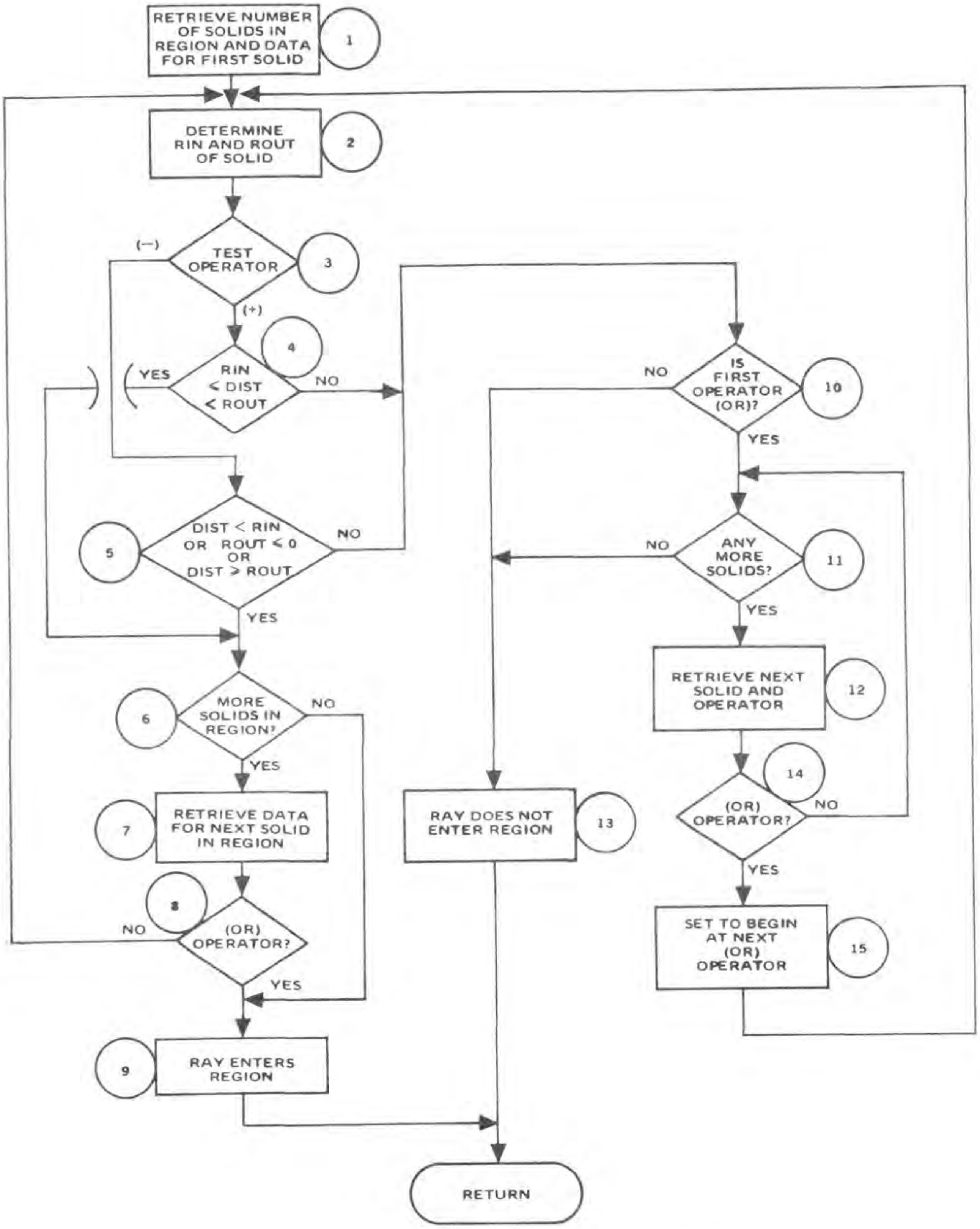

RETRIEVE NUMBER OF SOLIDS IN REGION AND DATA FOR FIRST SOLID
1
DETERMINE RIN AND ROUT OF SOLID
2
TEST OPERATOR
3
(−)
(+)
RIN < DIST < ROUT
4
YES
NO
DIST < RIN OR ROUT ≤ 0 OR DIST > ROUT
5
NO
YES
MORE SOLIDS IN REGION?
6
NO
YES
RETRIEVE DATA FOR NEXT SOLID IN REGION
7
(OR) OPERATOR?
8
NO
YES
RAY ENTERS REGION
9
IS FIRST OPERATOR (OR)?
10
NO
YES
ANY MORE SOLIDS?
11
NO
YES
RETRIEVE NEXT SOLID AND OPERATOR
12
RAY DOES NOT ENTER REGION
13
(OR) OPERATOR?
14
NO
YES
SET TO BEGIN AT NEXT (OR) OPERATOR
15
RETURN

FIG. 58. Subroutine WOWI Concept Flowchart

SYMBOLS AND ABBREVIATIONS

The symbols and abbreviations appearing in the Mathematical Model Section are defined in the following tables.

LIST OF SYMBOLS AND ABBREVIATIONS (MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
A	BB	Intermediate term used in ELL section to represent $(\overline{D}_1^2 - \overline{D}_2^2 - C^2)/ -2C$	ND*
A	COEF(1)	Coefficient of S^4 of quartic equation $AS^4 + BS^3 + CS^2 = DS + E = 0$	ND
A	---	Coefficient of X^2 of biquadratic equation of Subroutine QRTIC	ND
A	---	Intermediate term for solving r_2 of TOR	Inches
A_i	ASQ(3)	Square of one of length vectors of RAW	Inches^2
A_i	AA(I,1)	X coefficient of equation for ith face of ARB	ND
a	C(1)	Coefficient of X^3 of quartic equation	ND
a	A	Length of semi-major axis of ellipse, $a^2 = \overline{A} \cdot \overline{A}$	Inches
a	---	Distance between intersect projected onto $\overline{H}$ vector and end of semi-major axis of REC	Inches
$\overline{A}$	A(3)	Semi-major axis of REC	Inches
$\overline{A}$	AA(3)	Semi-major axis unit vector of base ellipse of TEC	ND
B	AA	Intermediate term used in ELL section to represent $(2\overline{D}_1 \cdot \overline{WB} - 2\overline{D}_2 \cdot \overline{WB})/-2C$	ND
B	COEF(2)	Coefficient of S^3 of quartic equation $AS^4 + BS^3 + CS^2 + DS + E = 0$	ND

*Non-dimensional

LIST OF SYMBOLS AND ABBREVIATIONS (MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
B	---	Coefficient of X of biquadratic equation of Subroutine QRTIC	ND
B	---	Intermediate term for solving r_2 of TOR	Inches
B	B	Coefficient of 2S term of quadratic equation $S^2+2BS+C=0$ for solving for intersect distances of sphere	Inches
B_i	AA(I,2)	Y coefficient of equation for ith face of ARB	ND
b	C(2)	Coefficient of X^2 of quartic equation	ND
b	---	Distance between intersect projected onto $\overline{H}$ vector and end of semi-minor axis of REC	Inches
b	B	Length of semi-minor axis of ellipse. $b^2 = \overline{B} \cdot \overline{B}$	Inches
$\overline{B}$	BB(3)	Semi-minor axis unit vector of base ellipse of TEC	ND
$\overline{B}$	B(3)	Semi-minor axis of REC	Inches
C	C	Distance between center of ellipse and a foci of REC	Inches
C	C	Length of ellipsoid major axis	Inches
C	C	Constant term of quadratic equation of sphere $S^2+2BS+C=0$ for solving intersect distances; equal to $(\overline{DX}^2 - R^2)$	Inches^2
C	---	Constant term of biquadratic equation of Subroutine QRTIC	ND

LIST OF SYMBOLS AND ABBREVIATIONS
(MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
C	COEF(3)	Coefficient of S^2 of quartic equation $AS^4 + BS^3 + CS^2 + DS + E = 0$	ND
C_i	AA(I,3)	Z coefficient of equation for ith face of ARB	ND
C_1	---	Intermediate term for solving coefficient of 2S of quadratic equation for TRC	Inches
C_2	RBRTVP	Intermediate term for solving constant term of quadratic equation for TRC	Inches
C	C(3)	Coefficient of X of quartic equation	ND
c	---	Distances from center point to focus of ellipse	Inches
$\overline{C}$	ASTER(IV)	Center coordinates of TOR	Inches
$\overline{C}$	TEMP(3)	Coordinates of center point	Inches
$\overline{CP}$	WA(3)	Coordinates of vector tangent at intersect point on surface of TRC or TEC	Inches
$\overline{CV}$	TEM1(3)	Vector from center of torus to center of circular plane of intersect	Inches
D	CELSIZ	Dimensions of grid cell	Inches
D	COEF(4)	Coefficient of S of quartic equation $AS^4+BS^3+CS^2+DS+E=0$	ND
D	TEM	Distance from vertex $\overline{V}$ to intersect projected onto $\overline{H}$ vector of given body	Inches
D_i	AA(I,4)	Constant term of equation for ith face of ARB	ND

LIST OF SYMBOLS AND ABBREVIATIONS (MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
d	C(4)	Constant term of quartic equation	ND
$\overline{DX}$	DX, DY, DZ	Vector from vertex $\overline{V}$ to origin of ray $\overline{XB}$	Inches
$\overline{D}_1$	D1X, D1Y, D1Z	x, y, and z coordinates of $\overline{XB}$ - (FOCI A) in ELL section	Inches
$\overline{D}_2$	D2X, D2Y, D2Z	x, y, and z coordinates of $\overline{XB}$ - (FOCI B) in ELL section	Inches
$\overline{d}$	---	Direction cosines of r_1 of TOR	ND
E	TERM	Constant term of quartic equation $AS^4 + BS^3 + CS^2 + DS + E = 0$	ND
$\overline{E}$	---	Vector from vertex to point of contact with side of RCC	Inches
f	---	Part of linear expression ex + f used in solving quartic equation	ND
$\overline{F}_a$	FOCIA(3)	x, y, and z coordinates of one of ellipsoid's foci	Inches
$\overline{F}_b$	FOCIB(3)	x, y, and z coordinates of one of ellipsoid's foci	Inches
$\overline{F1}$	F1	Intersect distance projected onto semi-major axis of TEC	Inches
$\overline{F2}$	F2	Intersect distance projected onto semi-minor axis of TEC	Inches
$\overline{F1}$	TEM(3)	Coordinates of a foci of REC	Inches
$\overline{F2}$	TEM1(3)	Coordinates of a foci of REC	Inches
G_i	G(3)	Dot product of direction cosines and one of length vectors of RAW	Inches
H	H	Horizontal distance from center of grid plane to random point within grid cell	Inches

LIST OF SYMBOLS AND ABBREVIATIONS
(MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
H_{ref}	HREF	Horizontal distance from center of grid plane to center of specified grid cell	Inches
H'	H	Horizontal distance from center of grid plane to lower left corner of specified grid cell	Inches
h	---	Ratio on $\overline{H}$ vector of height of hit on cylindrical surface of RCC	ND
$\overline{H}$	H(3)	x, y, and z coordinates of height vector of given body	Inches
$\overline{H}_1$, $\overline{H}_2$, $\overline{H}_3$	ASTER(JA)	x, y, and z coordinates of three mutually perpendicular length vectors of BOX or RAW	Inches
$\overline{h}$	---	Vector to point of intersect of intersection ellipse with $\overline{H}$ vector of TEC	Inches
I	II	Row number of grid plane	ND
I_h	IH	Random number for computing random horizontal point in given cell	ND
I_V	IV	Random number for computing random vertical point within given cell	ND
J	J	Column number of grid plane	ND
k	KK	Specific number of grid cell	ND
$\overline{L}$	TEM(3)	Coordinates of point on line from foci through intersect at distance equal to twice length of semi-major axis of REC	Inches
M	A	Length of semi-major axis of intersection ellipse of TEC	Inches
M	---	Center of specified grid cell	ND

LIST OF SYMBOLS AND ABBREVIATIONS (MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
M	B	Length of semi-minor axis of intersection ellipse of TEC	Inches
$\overline{m}$	---	Vector from midpoint of cross section to intersect of TOR	Inches
N	NEND	Total number of cells in grid plane	ND
N_x	NX	Number of horizontal cells in grid plane	ND
N_y	NY	Number of vertical cells in grid plane	ND
n	TLK	Length of normal to slanted side of RAW	Inches
$\overline{N}$	HN(3)	Normal unit vector of base ellipse of TEC	ND
$\overline{N}$	WB(3)	Direction cosines of vector at intersect of body	ND
$\overline{N}$	---	Normal vector to slanted surface of RAW	Inches
$\overline{N}$	ORMAL(3,I)	Normal unit vector to triangle surface in ARS	ND
$\overline{n}$	XN(3)	Unit normal vector of TOR	ND
P	---	Additional length of semi-major axis of intersect ellipse that is greater than semi-major axis of top ellipse of REC	Inches
P_i	PV(3)	Dot product of vector $(\overline{XB} - \overline{V})$ and one of length vectors of RAW	Inches^2
P1-P8	---	Eight points which describe the ARB	Inches

LIST OF SYMBOLS AND ABBREVIATIONS (MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
Q	XNOS	+1.0 or -1.0 multiplier for directing normal vector into (entry) or out of (exit) body at intersect	ND
$\overline{Q}$	---	Intersect on quadratic surface of TRC projected onto $\overline{H}$ vector	Inches
R	R	Radius of given body	Inches
R_B	RB	Radius of base plane of TRC	Inches
R_T	RT	Radius of top plane of TRC	Inches
RIN	RIN	Distance from ray origin to entry intersect of given body	Inches
ROUT	ROUT	Distance from ray origin to exit intersect of given body	Inches
R1	R1	Length of semi-major axis of base ellipse of TEC	Inches
R2	R2	Length of semi-minor axis of base ellipse of TEC	Inches
R3	R3	Length of semi-major axis of top ellipse of TEC	Inches
R4	R4	Length of semi-minor axis of top ellipse of TEC	Inches
r_1	R1	Major radius of TOR	Inches
r_2	R2	Radius of circular cross section of TOR	Inches
S	DIST	Scalar value of the distance from given ray point to next intersect	Inches
S_V	RIN, ROUT	Distance to intersect of side containing $\overline{V}$ of BOX or RAW	Inches

LIST OF SYMBOLS AND ABBREVIATIONS
(MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
S_{V+H}	RIN, ROUT	Distance to intersect of one of sides opposite $\overline{V}$ of BOX or RAW	Inches
T	---	Intermediate variable for computing semi-major axis of intersect ellipse of TEC	Inches
$\overline{T}$	TEM(3)	Coordinates of projected point of TRC or TEC	Inches
$\overline{U}$	---	ARS point of given triangle	Inches
V	V	Vertical distance from center of grid plane to random point within grid cell	Inches
V'	V	Vertical distance from center of grid plane to lower left corner of a specified cell	Inches
V_{ref}	VREF	Vertical distance from center of grid plane to center of specified grid cell	Inches
$\overline{V}$	V(3)	ARS point of given triangle	Inches
$\overline{V}$	V(3)	x, y, and z coordinates of vertex of given body	Inches
$\overline{VP}_A$	VPA	Dot product of semi-major axis vector and vector $(\overline{V} - \overline{XB})$ of REC	Inches
$\overline{VP}_B$	VPB	Dot product of semi-minor axis vector and vector $(\overline{V} - \overline{XB})$ of REC	Inches
$\overline{VP}_H$	VPH	Dot product of height vector and vector $(\overline{V} - \overline{XB})$ of REC	Inches
$\overline{VV}$	---	Vertex of intersection ellipse of TEC	Inches
$\overline{W}_x$, $\overline{W}_y$, $\overline{W}_z$	WB(3)	Direction cosines of ray	ND
$\overline{W}$	W(3)	ARS point of given triangle	Inches

LIST OF SYMBOLS AND ABBREVIATIONS
(MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
$\overline{W}_A$	WBA	Dot product of semi-major axis and direction cosines of ray for REC	Inches
$\overline{W}_B$	WBB	Dot product of semi-minor axis and direction cosines of ray for REC	Inches
$\overline{W}_H$	WH	Dot product of height vector and direction cosines of ray for REC	Inches
$\overline{WB}$	WB(3)	Direction cosines of ray	ND
$\overline{WI}$	WI(3)	Direction cosines of $\overline{H}$ vector	ND
$\overline{WN}$	WN(3)	Direction cosines of vector from $\overline{V}$ to $\overline{X}$ of given body	ND
$\overline{WS}$	WS(3)	Original direction cosines of ray from attack plane	ND
$\overline{WX}$	WN(3)	Direction cosines from focus to intersect on ellipse of REC	ND
$\overline{X}$	---	Intersect point on given body	Inches
$\overline{XB}$	XB	Starting point of ray	Inches
$\overline{XH}$	XP(3)	Coordinates of intersect point projected onto $\overline{H}$ vector of given body	Inches
$\overline{XI}$	XRY	Intersect coordinates on a given plane of RPP	Inches
$\overline{XP}$	S	Intersect point of given body	Inches
$\overline{XP}$	R1, R2, R3, R4	Point of intersect of TOR	Inches
$\overline{XS}_I$	XS(I)	Boundary coordinate for a given plane	Inches
$\overline{X}_1$	R1, R2, CP, CM	Vector to entry intersect of body	Inches

LIST OF SYMBOLS AND ABBREVIATIONS (MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
$\overline{X}_2$	R1, R2 CP, CM	Vector to exit intersect of body	Inches
$\overline{X}'$	R1, R2 CM, CP	Intersect point of REC and transformed ray	Inches
$\overline{X}_{min}$	XS(1)	Minimum X boundary coordinate of RPP	Inches
$\overline{X}_{max}$	XS(2)	Maximum X boundary coordinate of RPP	Inches
$\overline{X}$	---	Value of X coordinate on surface of ellipse for solving intersect with REC	Inches
x_o, y_o, z_o	XP(3)	Fixed point on ray shotline	Inches
$\overline{Y}_{min}$	XS(3)	Minimum y boundary coordinate of RPP	Inches
$\overline{Y}_{max}$	XS(4)	Maximum y boundary coordinate of RPP	Inches
y	---	Value of y coordinate on surface of ellipse for solving intersect with REC	Inches
$\overline{Z}_{min}$	XS(5)	Minimum z boundary coordinate of RPP	Inches
$\overline{Z}_{max}$	XS(6)	Maximum z boundary coordinate of RPP	Inches
α	---	Distance along ray from plane of base ellipse to plane of top ellipse of TEC	Inches
β	---	Distance along ray from start of ray to plane of base ellipse of TEC	Inches
γ	---	Ratio on $\overline{H}$ vector of height of hit on quadratic surface of TRC and TEC	ND

LIST OF SYMBOLS AND ABBREVIATIONS
(MATHEMATICAL MODEL)

Symbol or Abbreviation	Equivalent in Simulation Model	Definition	Units
λ	AMBDA	Coefficient of 2S from quadratic equation of body for solving distance S to intersect of quadratic surface	Inches
λ'	AMBD	Coefficient of 2S from quadratic equation $\tau S^2 - 2\lambda' S + \mu' = 0$ for TRC	ND
τ	DEN	Coefficient of S^2 from quadratic equation of body for solving distance S to intersect of quadratic surface	ND
θ	---	Ratio of distance between base ellipse and intersect ellipse to distance between two planar surfaces of TEC	ND
μ	UMU	Constant term from quadratic equation of body for solving distance S to intersect of quadratic surface	Inches
μ'	UM	Value of constant term from quadratic equation $\tau S^2 - 2\lambda' S + \mu' = 0$ for TRC	ND

SECTION III

SIMULATION MODEL

This section discusses the manner in which calculations are performed within the computer routines. The main routine controls the sequence of computations, making frequent use of subroutines for specific calculations. The program structure and subroutines' relationships are illustrated in Figure 59.

In the paragraphs following, the individual FORTRAN statements that make up the simulation model include references to Section II, Mathematical Model. These references consist primarily of the boxed equation that gave rise to the FORTRAN statement, but not its derivation. The same boxed equation can be found in the Mathematical Model with its derivation. The number enclosed in parentheses associated with each equation is its identifying number in Section II. In addition, comment statements appearing in the source listing are included in the simulation model.

COMMON and DIMENSION FORTRAN statements, whose purpose is to set aside arrays to store values of the variables appearing in the statements, are in several routines. The definitions of the variable names, as well as the variables in the subroutine call statements, are provided in the list of symbols and abbreviations at the end of this section.

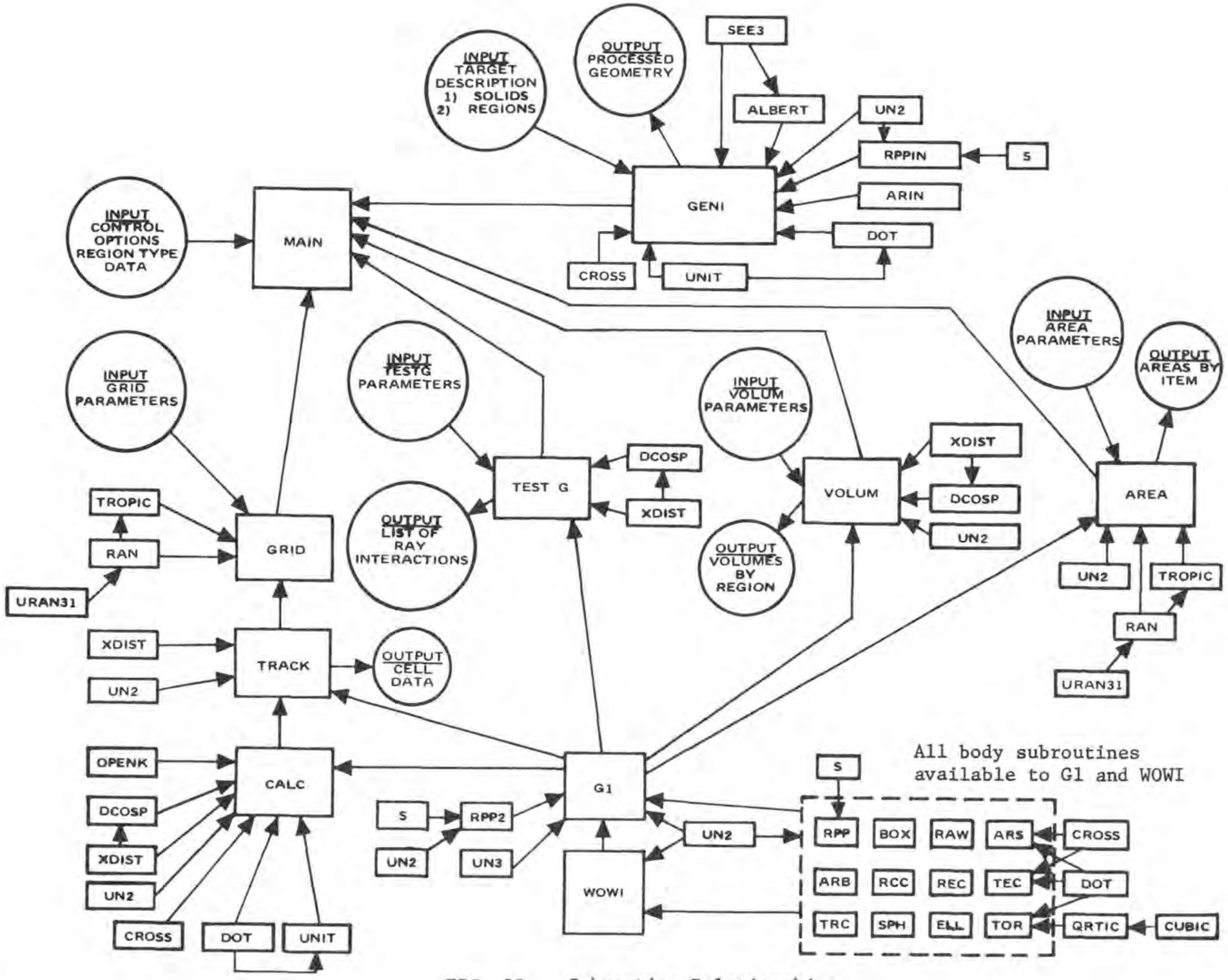


FIG. 59. Subroutine Relationships

PROGRAM MAGIC - MAIN ROUTINE

The purpose of the MAIN routine of the MAGIC program is to direct the entire flow of the MAGIC program, calling on various subprograms to perform specific functions.

There are essentially two phases to the MAGIC program: geometry input processing and main ray-tracing control. The MAIN routine initializes various program constants and parameters and enters from card input the program control parameters. The target geometry is entered by Subroutine GENI, which enters and prepares the target geometry data for the ray-tracing phase of the MAGIC program. If the target geometry data has already been processed by a previous run, the input tape is entered, and Subroutine GENI is therefore not called.

After the input processing phase, the MAIN program has the option, based on the input control parameters, of calling Subroutine TESTG to fire a ray from point to point and of calling Subroutine VOLUM to compute the volumes by region. The region identifiers and space code numbers are then entered and stored into the MASTER-ASTER array.

At this point, the ray-tracing phase of the program is initiated. A card is entered with the number of angles of attack, and Subroutine GRID is called to perform the ray-tracing control phase for each aspect angle. When the ray-tracing phase of the program is complete, the MAIN program has the option, based on the input control parameter, of calling Subroutine AREA to compute the presented area by component for each aspect angle.

The statements

```
DIMENSION A(6)
DIMENSION MASTER(10000)
```

are used to dimension the MASTER array for 10,000 words and to dimension six-element array A for input of alphameric region data.

The statements

```
COMMON ASTER(10000)
COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
COMMON/TEMPOR/XS(6),X(6),IX(8),IT(10),IA(9),IN(9)
COMMON/WALT/LIRFO,NG1ERR
COMMON/CONTRL/ITESTG,IRAYSK,IENTLV,IVOLUM,IWOT,ITAPE8,NO,IYES
COMMON/ENGEOM/LEGEOM
COMMON/SIZE/NDQ
```

```
      COMMON/ERR/IERR0
      COMMON/RANDM/IRANDM
```

are used to pass information into and out of this routine via the COMMON statements.

The statement

```
      EQUIVALENCE (ASTER,MASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statements

```
C
  901 FORMAT(1H1,32HTHIS IS THE 11 APR 69 VERSION OF /
     1          1H ,32HTHE BRLESC MAGIC PROGRAM  ****** //)
  902 FORMAT(16H BEGIN EXECUTION)
  903 FORMAT(8I10)
  904 FORMAT(1H0,10X,42HTHE TAPE 4 USED FOR THIS RUN HAS THE TITLE /
     1  10A6/)
  905 FORMAT(1H0,10HENTER GENI)
  906 FORMAT(1H0,12HLEAVING GENI)
  907 FORMAT(1H0,35HTERMINATION ON GEOMETRY INPUT ERROR,5X,5HIERR=,I5)
  908 FORMAT(1H1,15HTESTG IS CALLED)
  909 FORMAT(1H0,13HLEAVING TESTG)
  910 FORMAT(1H1,24HREGION TYPE DATA FOLLOWS, 8X,6HLIRFO=,I10/
     1          1H ,6HREGION,6X,4HCODE,6X,4HTYPE,6X,11HDESCRIPTION/)
  911 FORMAT(3I10,10X,6A6)
  912 FORMAT(I6,I10,I9,7X,6A6)
  913 FORMAT(1H0,23HNO ROOM FOR IDENT TABLE,5X,7HLEGEOM=,I7,5X,
     1     6HLIRFO=,I7)
  914 FORMAT(1H0,32HWRITE TAPE 1 OPTION IS SPECIFIED)
  915 FORMAT(I5,10X,10A6)
  916 FORMAT(1H1,11HENTER VOLUM)
  917 FORMAT(1H0,13HLEAVING VOLUM)
  918 FORMAT(1H ,6H 999.9)
  919 FORMAT(1H1,11HEND OF CASE,I5)
  925 FORMAT(1H1,32HNUM OF ASPECT ANGLES FOR GRID IS,I5)
  927 FORMAT(10I5)
  928 FORMAT(1H1,32HNUM OF ASPECT ANGLES FOR AREA IS,I5)
  929 FORMAT(1H0,31HNUMBER OF G1 ERRORS ENCOUNTERED,I5)
  930 FORMAT(1H0,31HNUMBER OF  0  ITEMS ENCOUNTERED,I5)
  999 FORMAT(1H0,10HEND OF RUN)
```

are used to format input data, output data, and output messages.

The statements

```
C
      IRANDM=0
      WRITE (6,901)
      WRITE (6,902)
```

are used to initialize variable IRANDM to zero for use by Function RAN for generating random numbers; and to print out the title and version of the program and a message that execution of the MAGIC program is beginning.

The statements

```
C
C1    INITIALIZE CONSTANTS
C
      I15=2**15
      I30=2**30
      PINF=1.0E50
      NO=0
      IYES=1
      IERR=0
      LBASE=1
      KLOOP=0
      NDQ=10000
```

are used to initialize constants I15 and I30 to 2^{15} and 2^{30}, respectively, for use in packing and unpacking data. PINF is set to an extremely large number to represent infinity. Variables NO and IYES are set to 0 and 1, respectively, for use in testing for various options during the MAGIC program. Error counter IERR is initialized to zero. Variable LBASE is set to one to represent the first storage location for data in the MASTER-ASTER array and is saved for printout of major pointer values by Subroutine GENI. KLOOP is initialized to zero and is used internally to keep track of the specific ray being traced. The constant NDQ is set to 10,000 to represent the word size of the MASTER-ASTER array.

The statements

```
C
C2    ENTER AND INITIALIZE OPTION PARAMETERS
C
      READ (5,903)IRDTP4,IWRTP4,ITESTG,IRAYSK,ICARDI,IENTLV,IVOLUM
      IF(IRDTP4.NE.0)IRDTP4=IYES
      IF(IWRTP4.NE.0)IWRTP4=IYES
      IF(ITESTG.NE.0)ITESTG=IYES
      IF(IRAYSK.NE.0)IRAYSK=IYES
      IF(ICARDI.NE.0)ICARDI=IYES
      IF(IENTLV.NE.0)IENTLV=IYES
      IF(IVOLUM.NE.0)IVOLUM=IYES
```

are used to enter the option data and to test each option for a not zero condition. Each variable that is not zero is set to equal IYES (IYES=1) for later option testing. Options follow:

IRDTP4 ≠ 0	Read target description from input tape 4 (binary tape)
IRDTP4 = 0	Enter and process target geometry via Subroutine GENI
IWRPT4 ≠ 0	Write target geometry description on tape 4 (binary tape) after entering and processing by Subroutine GENI
IWRPT4 = 0	Do not write target description on tape 4
ITESTG ≠ 0	Call Subroutine TESTG
ITESTG = 0	Do not call Subroutine TESTG
IRAYSK ≠ 0	Skip random number of grid cells during Subroutine GRID
IRAYSK = 0	Do not skip random number of grid cells during Subroutine GRID
ICARDI	Not used
IENTLV ≠ 0	The region enter/leave tables are to be printed out during Subroutine GENI
IENTLV = 0	Do not print out enter/leave tables
IVOLUM ≠ 0	Call Subroutine VOLUM
IVOLUM = 0	Do not call Subroutine VOLUM

The statements

```
C
C3    ENTER TARGET GEOMETRY FROM INPUT TAPE 4
C
      IF(IRDTP4.EQ.NO) GOTO 10
      READ (4) LBASE.LEGEOM.NDQ.(ASTER(L).L=1.NDQ).LBODY.LREGD.LRIN
     1  LROT.LIO.LIRFO.NRPP.NBODY.NRMAX.PINF.IT
```

```
      WRITE (6,904)(IT(I),I=1,10)
      GOTO 20
```

are used to determine if the target description is to be entered from input tape 4. If it is, the data is entered from tape 4, a message is written out with the title of tape 4, and the program branches around the section for entering and processing the data by Subroutine GENI.

The statements

```
C
C4    CLEAR MASTER-ASTER ARRAY
C
   10 DO 11 I=LBASE,NDQ
      ASTER(I)=0.
   11 CONTINUE
```

are executed when the target geometry is not entered by tape 4 but is to be entered and processed by Subroutine GENI. These statements zero the 10,000 words that are to be the MASTER-ASTER array.

The statements

```
C
C5    ENTER AND PROCESS TARGET GEOMETRY VIA SUBROUTINE GENI
C
      WRITE (6,905)
      CALL GENI
      WRITE (6,906)
      IF(IERR.LE.0)GOTO 12
      WRITE (6,907)IERR
      STOP
```

are used to output the message ENTER GENI and to call Subroutine GENI to enter and process the target description from card input. When Subroutine GENI returns control, the message LEAVING GENI is printed. Error counter IERR is tested to determine if any errors occurred in Subroutine GENI. If errors occurred, the message TERMINATION ON GEOMETRY INPUT ERROR and the number of errors are printed, and the program is then terminated by the STOP statement.

The statements

```
C
C6     WRITE OUT TARGET GEOMETRY TO OUTPUT TAPE 4
C
   12 IF(IWRTP4.EQ.NO)GOTO 20
      WRITE (4) LBASE,LEGEOM,NDQ,(ASTER(L),L=1,NDQ),LBODY,LREGD,LRIN,
     1  LROT,LIO,LIRFO,NRPP,NBODY,NRMAX,PINF,IT
```

are executed if Subroutine GENI was called and if no errors occurred during Subroutine GENI. These statements are used to write out the target geometry description on tape 4 if control word IWRTP4 is set to 1.

The statements

```
C
C7     CALL SUBROUTINE TESTG
C
   20 IF(ITESTG.EQ.NO)GOTO 30
      WRITE (6,908)
      CALL TESTG
      WRITE (6,909)
      ITESTG=NO
```

are used to determine if Subroutine TESTG is to be called (ITESTG=1). If it is, the message TESTG IS CALLED is printed, and Subroutine TESTG is called. When control is returned from Subroutine TESTG, the message LEAVING TESTG is printed, and the ITESTG option variable is set to zero.

The statements

```
C
C8     CALL SUBROUTINE VOLUM
C
   30 IF(IVOLUM.EQ.NO)GOTO 40
      WRITE (6,916)
      CALL VOLUM
      WRITE (6,917)
      IVOLUM=NO
```

are used to determine if Subroutine VOLUM is to be called (IVOLUM=1). If it is, the message ENTER VOLUM is printed, and Subroutine VOLUM is called. When control is returned from Subroutine VOLUM, the message LEAVING VOLUM is printed, and the IVOLUM option variable is set to zero.

The statements

```
C
C9     REGION IDENTIFICATION DATA        FORMAT - / ICODE / IDENT /
C           IRN    = REGION NUMBER
C           ICODE = ITEM CODE
C           IDENT = SPACE CODE AND SPECIAL IDENTIFICATION
C                      SPECIAL IDENTIFICATION = 10,20,30,40,50,60,70,80,90
C                        NO IDENT CODE=0  SKIRT=10  ARMOR=20  TARGET=30
C                      SPACE CODES    EXTERIOR VOLUME = 1
C                                     INTERIOR VOLUME = -1,1-9,...,91-99
C
C
   40 LIRFO=NDQ-NRMAX-10
      IF(LIRFO.GT.LEGEOM)GOTO 41
      WRITE(6,913)LEGEOM,LIRFO
      STOP
   41 WRITE (6,910)LIRFO
```

are used to compute a location, LIRFO, near the end of the MASTER-ASTER array where the region type data is to be stored. After computing the beginning location, based on the number of regions in the target geometry, a test is made to determine if this location occurs after the last location of the target geometry data entered by Subroutine GENI. If LIRFO occurs before the end of the geometry data, an error message with the value of LIRFO and the value of the last location of geometry is printed. The program is then terminated by the STOP statement. If LIRFO occurs after the end of the geometry data, a message and the region table headings are printed.

The statements

```
C
C10    ENTER AND STORE REGION ID DATA
C
   42 READ(5,911)IRN,ICODE,IDENT,(A(I),I=1,6)
      IF(IRN.LE.0)GOTO 50
      WRITE (6,912)IRN,ICODE,IDENT,(A(I),I=1,6)
      IDENT=IDENT+1
      K=LIRFO+IRN-1
      MASTER(K)=ICODE*I15+IDENT
      GOTO 42
```

are used to enter from card input the region number, component code, space code and special identification, and a 36-letter description of the region. A test is made on the region number to determine if all of the region data cards are entered. If not, the data on the input card is written out.

A one is added to IDENT to prevent storing a negative number, the location in the MASTER array is computed based on the region number, and the component code and space code are packed in the MASTER array. The program then branches to enter the next region data card.

The statements

```
C
C11         NOAA   = NUMBER OF ASPECT ANGLES FOR SUBROUTINE GRID
C           ITAPE8 = SUPPRESS PRINTER OPTION
C           IWOT   = WRITE OPTION FOR TAPE 1
C           NAREA  = NUMBER OF ASPECT ANGLES FOR SUBROUTINE AREA
C
   50 READ (5,927)NOAA,IWOT,ITAPE8,NAREA
      IF(IWOT.NE.0)IWOT=IYES
      IF(ITAPE8.EQ.0)GOTO 51
      ITAPE8=NO
      GOTO 52
   51 ITAPE8=IYES
   52 IF(IWOT.EQ.NO)GOTO 60
      REWIND 1
      WRITE (6,914)
      WRITE(1,915)NOAA,(IT(I),I=1,10)
```

are used to enter NOAA, the number of aspect angles for Subroutine GRID; IWOT the write option for output tape 1; ITAPE8, the suppress printer option for Subroutines GRID and TRACK; and NAREA, the number of aspect angles for Subroutine AREA. If IWOT is not equal to zero, it is set to one. If ITAPE8 is equal to zero, the printer is not to be suppressed. IWOT is then tested for a zero or one value. If IWOT=1, output tape 1 is rewound, a message is printed out that the write tape 1 option was specified, and a 60-column alphanumeric title of the problem is written out on tape 1 along with the number of aspect angles to be processed by Subroutine GRID

The statements

```
C
   60 IF(NOAA.LE.0)GOTO 70
      WRITE (6,925)NOAA
```

are used to determine if there are any aspect angles to be processed by Subroutine GRID. If not, the program branches to test the Subroutine AREA option. If there are aspect angles to be processed, the number of aspect angles to be processed by Subroutine GRID is written out.

The statements

```
C
C12   CALL SUBROUTINE GRID FOR EACH ASPECT ANGLE
C
      DO 61 I=1,NOAA
      IERR=0
      IERR0=0
      CALL GRID
      IF(IWOT.EQ.IYES)WRITE(1,918)
      WRITE (6,919)I
      WRITE (6,929)IERR
      WRITE (6,930)IERR0
   61 CONTINUE
C
```

consist of a DO loop which is used to perform the ray-tracing phase of the MAGIC program by calling Subroutine GRID for each aspect angle. When control is returned after Subroutine GRID processes each aspect angle, the write output tape 1 option, IWOT, is tested for an equal-to-one condition. If IWOT is equal to one, the Hollerith code 999.9 is printed on output tape 1; a message and the aspect angle number are printed; the number of Subroutine G1 errors is printed; and the number of 0 items encountered is printed. The Subroutine G1 error counter and the number of 0 items counter are set to zero before continuing to the next aspect angle.

The statements

```
   70 IF(NAREA.LE.0)GOTO 99
      WRITE (6,928)NAREA
```

are executed after all aspect angles (if any) for Subroutine GRID have been processed. These statements are used to determine if there are any aspect angles for Subroutine AREA to be processed. If there are, the number of aspect angles for Subroutine AREA is printed.

The statements

```
C
C13   CALL SUBROUTINE AREA FOR EACH ASPECT ANGLE
C
      DO 71 I=1,NAREA
      IERR=0
      CALL AREA
      WRITE (6,919)I
   71 CONTINUE
```

consist of a DO loop which is used to call Subroutine AREA for each aspect angle to be processed. When control is returned after Subroutine AREA processes each aspect angle, a message with the aspect angle number is printed; and the Subroutine G1 error count is set to zero before continuing with the next aspect angle.

The statements

```
C
   99 WRITE (6,999)
      STOP
```

are executed when all processing in the MAGIC program is complete. These statements are used to print out the message END OF RUN and to terminate the program with the STOP statement.

Subroutine GENI

Subroutine GENI is the main control routine for target geometry input processing. Its purpose is to organize the target geometry input data into a usable format for subsequent subroutines. All of the organized data is stored in a large array called the MASTER-ASTER array.

The processing of the body input descriptions is accomplished by calling Subroutine RPPIN to enter and process the RPP data. All other bodies are processed directly by Subroutine GENI, with the exceptions of the arbitrary polyhedron (ARB), which is processed by calling Subroutine ALBERT when input for an ARB is encountered; and the arbitrary surface (ARS), which is processed by calling Subroutine ARIN when input for an ARS is encountered.

When all of the body input data has been processed, Subroutine GENI enters the input data for the region descriptions, stores it in the MASTER-ASTER array, and then computes the region enter/leave tables.

During all of the input processing, Subroutine GENI also checks for errors in the input data and prints out appropriate diagnostic error messages.

The statements

```
 DIMENSION ITY(11),IAN(8),IAA(8),FX(20),NBOD(11),
1  NO0(3),NO1(3),NO2(3),O4(3),TT(3),TT1(3),TT2(3)
 DIMENSION MASTER(10000)
 COMMON ASTER(10000)
 COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
 COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
 COMMON/TEMPOR/XS(6),X(6),IX(8),IT(10),IA(9),IN(9)
 COMMON/CONTRL/ITESTG,IRAYSK,IENTLV,IVOLUM,IWOT,ITAPE8,NO,IYES
 COMMON/SIZE/NDQ
 COMMON/UNCLE/NN,IC(4)
 COMMON/RRPP/LRPPD,LABUT
 COMMON/ENGEOM/LEGEOM
```

are used to dimension arrays and to pass information into and out of this subroutine.

The statement

```
EQUIVALENCE(ASTER,MASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statements

```
C
  901 FORMAT(1H0,24HSTART READING SOLID DATA)
  902 FORMAT(10A6)
  903 FORMAT(1H0,10A6/)
  904 FORMAT(7I10)
  905 FORMAT(4X,34HNO. OF RECTANGULAR PARALLELEPIPEDS,I10/
     1          4X,34HNO. OF SOLIDS                              ,I10/
     2          4X,34HMAX NO. OF REGIONS                         ,I10)
  906 FORMAT(1H0,45X,32HRECTANGULAR PARALLELEPIPED INPUT)
  911 FORMAT(1H0,50X,22HDESCRIPTION OF  SOLIDS)
  912 FORMAT(3A1,A3,A4,6F10.5)
  913 FORMAT(1H0,6HITYPE ,A3,27H DOES NOT MATCH WITH AN ITY)
  914 FORMAT(I9,1X,3A1,3X,A3,A4,3X,8I5)
  915 FORMAT(I8,1X,3A1,2X,A3,A4,6X,6F12.5)
  916 FORMAT(25X,6F12.5)
  917 FORMAT(1H0,38HNO MORE ROOM FOR SOLID DATA     LOATA=,I10,
     1  5X,5HLBOT=,I10,5X,4HNDQ=,I10)
  918 FORMAT(1H0,25HFINISH READING SOLID DATA)
  919 FORMAT(1H0,  5HLREGD,7H  LREGL,7H  LENLV,7H   LRIN,7H   LROT,
     1     7H    LIO,7H LEGEOM/I5,6I7)
  920 FORMAT(1H1,36X,23HREGION COMBINATION DATA)
  921 FORMAT(I5,1X,9(A2,I5))
  922 FORMAT(1H0,30HERROR IN DESCRIPTION OF REGION,I5,
     19H IN FIELD,I2,5X,24HBODY NUM.GT.NRPP + NBODY)
  923 FORMAT(10X,9(1H(,A2,I5,1H),1X))
  924 FORMAT(I8,2X,9(1H(,A2,I5,1H),1X))
  925 FORMAT(1H0,30HILLEGAL OPERATOR IN ABOVE CARD,5X,A2,
     1  9H IN FIELD,I2)
  926 FORMAT(1H0,29HERROR IN REGION INPUT    IR=,I5,14H OR N.GT.NRMAX)
  927 FORMAT(1H0,39HNO MORE ROOM FOR REGION DATA     LOATA=,I10,
     1  5X,4HNDQ=,I10)
  928 FORMAT(1H0,26HFINISH READING REGION DATA)
  929 FORMAT(14H ERROR, REGION,I10,18H IS PART OF REGION,I10)
  930 FORMAT(24H FINISH CHECKING REGION ,I5)
  931 FORMAT(1H0,34HNO MORE ROOM FOR ENTER LEAVE TABLE,5X,
     1  6HLDATA=,I10,5X,4HNDQ=I10,5X,4HPASS,I2,5X,3HIR=,I10)
  932 FORMAT(1H0,28HTOTAL ROOM FOR GEOMETRY DATA,5X,7HLEGEOM=,I6)
  933 FORMAT(1H0,5HENTER,18I6/(23X,15I6))
  934 FORMAT(1H ,5HLEAVE,18I6/(23X,15I6))
  935 FORMAT(1H1,50X,18HBEGIN ARRAY OUTPUT/)
  936 FORMAT(3(3I6,1X,E11.4,3H $ ))
  937 FORMAT(/)
  938 FORMAT(1H0,34HFINISH A PASS OF ENTER LEAVE TABLE,I5)
  940 FORMAT(10X,6F10.5)
  941 FORMAT(1H0,37HTERMINATION ON BAD REGION DESCRIPTION)
  942 FORMAT(1H0,32HERROR IN DESCRIPTION OF BODY NUM,I6/
     1     7H VECTOR,3F12.5,24H IS NOT PERPENDICULAR TO /
     2     7H VECTOR,3F12.5/)
  943 FORMAT(1H0,27HERROR IN DESCRIPTION OF TOR,5X,8HR2.GT.R1/)
  944 FORMAT(1H0,27HERROR IN DESCRIPTION OF TRC,5X,7HR1 = R2/)
  945 FORMAT(1H0,5HLBASE,7H  LRPPD,
     1     7H  LABUT,7H  LBODY,7H   LBOD,7H  LDATA,7H   LBOT,7H  LSCAL,
     2     7H  LTRIP,7H    NDQ/I5,9I7)
  946 FORMAT(1H1,17HENTER-LEAVE TABLE)
  947 FORMAT(1H0,11(2X,A3)/11I5)
  948 FORMAT(1H0,27HERROR IN DESCRIPTION OF TEC,5X,
     1     41HHEIGHT VECTOR IS PARALLEL TO BASE ELLIPSE)
```

are used to format the input data, output data, and output messages.

The statements

```
 DATA ITY(1),ITY(2),ITY(3),ITY(4),ITY(5),ITY(6),ITY(7),ITY(8)
1    / 3HBOX, 3HSPH, 3HRCC, 3HREC, 3HTRC, 3HELL, 3HRAW, 3HARB /
 DATA ITY(9),ITY(10),ITY(11)
1    / 3HTEC, 3HTOR , 3HARS /
 DATA IAA(1),IAA(2),IAA(3),IAA(4),IAA(5),IAA(6),IAA(7),IAA(8)
1    / 2H   , 2HOR , 2H R ,2HR  , 2HRA , 2HAR , 2H A , 2HA  /
 DATA IAN(1),IAN(2),IAN(3),IAN(4),IAN(5),IAN(6),IAN(7),IAN(8)
1    /    4 ,    1 ,    1 ,    1 ,    2 ,    2 ,    3 ,    3 /
 DATA IBL/1H /
```

are used to assign abbreviated forms of the 11 bodies used to describe the target geometry to array ITY. Logical operator evaluation data is entered into array IAA, and integers are placed in the IAN array for converting the logical operator of a body to a number. A blank is entered into IBL for control purposes.

The statements

```
   DO 10 I=1,11
10 NBOD(I)=0
```

are used to clear the NBOD array which is used as a counter for each body type used to describe the target geometry.

The statements

```
C
C2    ENTER AND PRINT OUT TITLE OF THE PROBLEM
C
      WRITE (6,901)
      READ(5,902)(IT(I),I=1,10)
      WRITE  (6,903)(IT(I),I=1,10)
```

are used to output the message START READING SOLID DATA and to enter and print out a 60-column alphanumeric title of the problem which will appear in the printed output.

The statements

```
C
C3    ENTER AND PRINT OUT THE PROGRAM CONTROL PARAMETERS
C
      READ(5,904)NRPP,NTRIP,NSCAL,NBODY,NRMAX,IPRIN,IRCHEK
      WRITE(6,905)NRPP,NBODY,NRMAX
```

are used to enter and print out the control parameters of the problem. Those values entered are NRPP, the number of rectangular parallelepipeds; NTRIP and NSCAL, dummy variables; NBODY, the number of different bodies used to describe the target geometry other than RPP's; NRMAX, the maximum number of regions made by the RPP's and other geometric configurations; IPRIN, a printing option (if IPRIN is other than zero, the MASTER-ASTER array will be printed out upon completion of Subroutine GENI); and IRCHEK, to check the validity of the region data if IRCHEK is other than zero.

The statements

```
C
C4    RPP
C
      WRITE (6,906)
      LAR=1
```

are used to output the message RECTANGULAR PARALLELEPIPED INPUT in preparation for reading the RPP input data into the MASTER-ASTER array. The variable LAR is then set to one to indicate the next available location in the MASTER-ASTER array.

The statements

```
C
C5    RPP DATA INPUT
C
      IF(NRPP.LE.0)GOTO 20
      CALL RPPIN(LAR)
      IF(IERR.GT.0)RETURN
```

test to determine if there is any RPP data to be processed. If not, control is transferred to compute pointers to the location of various body data in the ASTER array. If there is RPP data to be processed, Subroutine RPPIN is called to enter the RPP data. When Subroutine RPPIN completes its task it returns with the variable LAR set to the next available location in the ASTER array. A test determines if there were any errors in the RPP input data. If there were errors, the subroutine returns to the MAIN program.

The statements

```
C
C7     LBODY STORAGE       RESERVE 3*(NRPP+NBODY) WORDS
C        /         BODY NUMBER          / LOC OF POINTER TO BODY DATA/
C        / REGION ENTER TABLE POINTER / REGION LEAVE TABLE POINTER /
C        / NUM REGIONS IN ENTER TABLE / NUM REGIONS IN LEAVE TABLE /
C
   20 LBOT=NDQ-2
      L=LAR
      LBODY=L+1
      LDATA=LBODY+3*(NBODY+NRPP)
      LBOD=LDATA
```

are used to set LBOT to the first location for storing body dimensions (stored backward from the end of the MASTER-ASTER array) and to equate L to the last location at the end of the RPP data in the ASTER array. LBODY is the beginning location of the body data pointers and is the next location after L. Pointer LBODY is saved for later printout of major pointer values. The beginning location of the body data is computed and assigned to LDATA, and LBOD is equated to LDATA for later printout of major pointer values.

The statement

```
C
   50 WRITE (6,911)
```

is used to output the message DESCRIPTION OF SOLIDS prior to processing the individual body data.

The statement

```
C
C10    ENTER DATA FOR BODY
C
       DO 370 N=1,NBODY
```

is used to begin a DO loop which will enter, process, and organize data for each geometric shape (descriptor) in the target geometry for use by the other subroutines in the MAGIC program.

The statements

```
       NN=N+NRPP
       LS1=0
```

are used to compute a value for NN based on the body number and the number of RPP's used to describe the target geometry and to initialize variable LS1 to zero. LS1 is used by Subroutine SEE3 for storing either triplet or scalar data into the ASTER array. LS1 = 0 means that triplet data is to be stored. LS1 = 1 means that scalar data is to be stored.

The statement

```
      READ(5,912) IC(1),IC(2),IC(3),ITYPE,IC(4),(FX(K),K=1,6)
```

is used to enter program control data [IC(1) and IC(4)], the body type (ITYPE), and the body dimensions.

The statements

```
      DO 51 I=1,11
      IF(ITYPE.EQ.ITY(I))GOTO 52
   51 CONTINUE
```

consist of a DO loop which compares each of the eleven possible body types with the body type entered until the body types match.

The statements

```
      WRITE (6,913)ITYPE
      STOP
```

are executed if no match is established between the eleven body types in the ITY array and body type entered. A message is printed out giving the body type entered and stating that there was no match with the body types in the ITY array. The program is then terminated.

The statements

```
   52 ITYPE=I
      NBOD(I)=NBOD(I)+1
      K=LBODY+3*(NRPP+N-1)
      MASTER(K)=ITYPE*115+LDATA
```

are used to assign the body type number (1-11) to ITYPE and to increment by one the number at NBOD referenced by the present body type number. A location in the body data section (LBODY) is computed from the number of RPP's in the target geometry and the number of the descriptor now being processed. The body type number and the pointer to the body data pointers are packed into the word at the location referenced by K.

The statement

```
C
C           BOX SPH RCC REC TRC ELL RAW ARB TEC TOR ARS
  200 GOTO(201,220,202,201,203,202,201,230,204,203,240),ITYPE
```

is executed if it was previously determined that the input contains all or part of the triplet and scalar data (dimensions) of the present body. This statement causes a branch to the appropriate section for the present descriptor.

The statements

```
  201 LE=12
      GOTO 210
  202 LE= 7
      GOTO 210
  203 LE= 8
      GOTO 210
  204 LE=13
```

are used to assign to variable LE the number of data elements required for the dimensions of the present body. LE is used in the following READ statement as the upper limit of the implied DO loop to enter the data elements required in addition to the six data elements in array FX. For a sphere, arbitrary polyhedron, or arbitrary surface, the program branches around these statements since all of the necessary data elements for a sphere are contained in array FX, and the ARB and ARS require special subroutines to input their data elements.

The statements

```
  210 WRITE (6,915)NN,IC(1),IC(2),IC(3),ITY(ITYPE),IC(4),(FX(J),J=1,6)
      READ(5,940)(FX(J),J=7,LE)
      WRITE (6,916)(FX(J),J=7,LE)
```

are used to write out the pertinent program data and the data elements for the body. The additional data elements required are then entered and written out.

The statement

```
C           BOX SPH RCC REC TRC ELL RAW ARB TEC TOR ARS
      GOTO(290,300,300,290,285,270,290,300,260,250,300),ITYPE
```

is used to cause a transfer to the appropriate body section to perform checks on the data or to compute additional data that is required that was not part of the input.

The statements

```
C          SPH
  220 WRITE (6,915)NN,IC(1),IC(2),IC(3),ITY(ITYPE),IC(4),(FX(J),J=1,4)
      GOTO 300
```

are used to write out the pertinent program data and the data elements if the present descriptor is a sphere with all of the data elements in array FX. The program then branches to process and store the data.

The statements

```
C
C14   ENTER BODY DATA FOR ARB
C
  230 WRITE (6,915)NN,IC(1),IC(2),IC(3),ITY(ITYPE),IC(4),(FX(J),J=1,6)
      CALL ALBERT(FX,LBOT,NDQ,LS1)
      GOTO 360
```

are used to write out the pertinent program data and the information on the card previously entered when the descriptor is an ARB. Subroutine ALBERT is then called to compute the data elements for the ARB from the face points supplied in array FX.

The statements

```
C
C15   ENTER BODY DATA FOR ARS
C
  240 CALL ARIN(LBOT,LDATA)
      GOTO 360
```

are executed if the body is an ARS and are used to call Subroutine ARIN to compute and store the data elements in the ASTER array.

The statements

```
C
C16        TOR      CONVERT NORMAL VECTOR TO UNIT VECTOR
C
  250 TT(1)=FX(4)
      TT(2)=FX(5)
      TT(3)=FX(6)
      CALL UNIT(TT)
      FX(4)=TT(1)
      FX(5)=TT(2)
      FX(6)=TT(3)
```

are executed when the body is a torus and the data elements were contained in array FX. These statements are used to compute additional information required to completely describe the torus. Specifically, these statements convert the coordinates of the normal vector of the torus from array FX to a unit vector by calling Subroutine UNIT. The resulting coordinates are then stored in array FX.

The statements

```
      IF(FX(7).GE.FX(8))GOTO 280
      WRITE (6,943)
      IERR=IERR+1
      GOTO 280
```

are used to compare the major and minor radii of the torus. If the minor radius is larger, an error message is printed out and the error count is incremented by one.

The statements

```
C
C17        TEC      VERIFY SEMI-MAJOR AND SEMI-MINOR AXES PERPENDICULAR
C
  260 FX(15)=FX(13)
      LE=15
      TT1(1)=FX(7)
      TT1(2)=FX(8)
      TT1(3)=FX(9)
      TT2(1)=FX(10)
      TT2(2)=FX(11)
      TT2(3)=FX(12)
```

are executed when the body is a truncated elliptic cone and the data elements are contained in array FX. These statements are used to move the value of the ratio of the larger to the smaller ellipse from FX(13), where it was entered, to FX(15). The variable LE is set to 15 for later use as the upper limit in an implied DO loop when the dimensions for the TEC are written out. The coordinates of the semi-major axis and semi-minor axis vectors are then stored in arrays TT1 and TT2, respectively.

The statements

```
      IF(ABS(DOT(TT1,TT2)).LE.0.01) GOTO 265
      WRITE (6,942)NN,TT1,TT2
      IERR=IERR+1
```

are used to determine if the absolute value of the dot product of the semi-major axis vector and semi-minor axis vector is less than a small value, which indicates that the two vectors are perpendicular, as they should be. If they are not perpendicular, an error message is written out and the error count is incremented by one.

The statements

```
C
C18    COMPUTE SEMI-MAJOR AXIS LENGTH AND CONVERT
C      SEMI-MAJOR AXIS TO UNIT VECTOR
C
  265 FX(13)=SQRT(DOT(TT1,TT1))
      CALL UNIT(TT1)
      FX(10)=TT1(1)
      FX(11)=TT1(2)
      FX(12)=TT1(3)
```

are used to compute the radius of the semi-major axis from the scalar value of the semi-major axis and store the result in FX(13). Subroutine UNIT is called to compute the unit vector of the semi-major axis and the resulting coordinates are stored in FX(10), FX(11), and FX(12).

The statements

```
C
C19    COMPUTE SEMI-MINOR AXIS LENGTH AND CONVERT
C      SEMI-MINOR AXIS TO UNIT VECTOR
C
      FX(14)=SQRT(DOT(TT2,TT2))
      CALL CROSS(TT,TT1,TT2)
      CALL UNIT(TT)
```

are used to compute the radius of the semi-minor axis from the scalar value of the semi-minor axis and store the result in FX(14). Subroutine CROSS is then called to compute the cross product of the semi-major axis vector and the semi-minor axis vector which results in a vector perpendicular to the base ellipse. Subroutine UNIT is called to convert the perpendicular vector to a unit vector, or the direction cosines of the normal to the base ellipse.

The statements

```
      HON=FX(4)*TT(1)+FX(5)*TT(2)+FX(6)*TT(3)
      IF(HON)267,266,268
```

are used to compute the dot product of the height vector and the normal unit vector to the base ellipse and to then determine the sign of the result.

The statements

```
 266 WRITE(6,948)
     IERR=IERR+1
     GOTO 268
```

are executed if the result of the above dot product was zero, which indicates that the height vector is parallel to the base plane. An error message is therefore printed out, and the error count is incremented by one.

The statements

```
 267 TT(1)=-TT(1)
     TT(2)=-TT(2)
     TT(3)=-TT(3)
 268 FX(7)=TT(1)
     FX(8)=TT(2)
     FX(9)=TT(3)
     GOTO 280
```

are used to store the coordinates of the normal to the base ellipse in FX(7), FX(8), and FX(9) if the dot product was positive, or after the direction cosines have been reversed if the dot product was negative.

The statement

```
C
C20   COMPUTE FOCI FOR ELL
C
 270 IF(IC(4).EQ.IBL)GOTO 300
```

is used to test location IC(4), previously entered, to determine if it contains blanks. Blanks in IC(4) indicate that FX already contains the coordinates of the foci and the length of the major axis of an ELL when the card containing FX was originally entered. Non-blank characters in IC(4) indicate that FX contains the coordinates of the center of the ELL and the coordinates of one end of the ELL, along with the radius of the minor axis of the ELL. Therefore, the coordinates of the foci and the length of the major axis must be computed.

The statements

```
     ASQ=FX(4)*FX(4)+FX(5)*FX(5)+FX(6)*FX(6)
     C=SQRT(ASQ-FX(7)*FX(7))
     A=SQRT(ASQ)
     FX(7)=A+A
```

are used to compute the distance from the center to a focus, the distance from the center to the end of the ELL along the major axis, and the length of the major axis, which is then stored in FX(7).

The statements

```
C
C21    COMPUTE X,Y,Z COMPONENTS OF FOCI
C
       CX=C*FX(4)/A
       CY=C*FX(5)/A
       CZ=C*FX(6)/A
```

are used to compute the x, y, and z components of the focus with respect to the center of the ELL.

The statements

```
C
C22    COMPUTE X,Y,Z COORDINATES OF FOCI
C
       FX(4)=FX(1)+CX
       FX(5)=FX(2)+CY
       FX(6)=FX(3)+CZ
       FX(1)=FX(1)-CX
       FX(2)=FX(2)-CY
       FX(3)=FX(3)-CZ
```

are used to compute the coordinates of the two foci and store the results in FX.

The statements

```
C
C23    PRINT OUT NEW INPUT
C
  280 WRITE (6,915)NN,IC(1),IC(2),IC(3),ITY(ITYPE),IC(4),(FX(J),J=1,6)
      WRITE (6,916)(FX(J),J=7,LE)
      GOTO 300
```

are executed when some of the data elements for the torus, the truncated elliptic cone, or the ellipsoid of revolution were not entered initially and had to be computed. These statements, therefore, are used to write out the pertinent program data and dimensions of the present descriptor if it is any one of the above three bodies.

The statements

```
C
C24         TRC    VERIFY LOWER AND UPPER RADII NOT EQUAL
C
  285 IF(FX(7).NE.FX(8))GOTO 300
```

```
      WRITE (6,944)
      IERR=IERR+1
      GOTO 300
```

are executed if the present descriptor is a truncated right cone and the dimensions (not pointers) are contained in array FX. These statements are used to compare the radii of the lower and upper bases to verify that they are unequal. If the radii are equal, an error message is written out and the error count is incremented by one.

The statements

```
C
C25   VERIFY THAT VECTORS ARE PERPENDICULAR IF BOX, RAW, OR REC
C
  290 IF(ABS(FX(4)*FX(7)+FX(5)*FX(8)+FX(6)*FX(9)).LE.0.01)GOTO 291
      WRITE (6,942)NN,(FX(J),J=4,9)
      IERR=IERR+1
```

are used to determine if the $\bar{H}1$ and $\bar{H}2$ vectors are perpendicular for a BOX or RAW, or if the height vector and the major axis vector are perpendicular for a REC. If they are not perpendicular, an error message and the pertinent vector coordinates are printed out.

The statements

```
  291 IF(ABS(FX(4)*FX(10)+FX(5)*FX(11)+FX(6)*FX(12)).LE.0.01)GOTO 292
      WRITE (6,942)NN,FX(4),FX(5),FX(6),FX(10),FX(11),FX(12)
      IERR=IERR+1
```

are used to determine if the $\bar{H}1$ and $\bar{H}3$ vectors are perpendicular for a BOX or RAW, or if the height vector and the minor axis vector are perpendicular for a REC. If they are not perpendicular, an error message and the pertinent vector coordinates are printed out.

The statements

```
  292 IF(ABS(FX(7)*FX(10)+FX(8)*FX(11)+FX(9)*FX(12)).LE.0.01)GOTO 300
      WRITE (6,942)NN,(FX(J),J=7,12)
      IERR=IERR+1
```

are used to determine if the $\bar{H}2$ and $\bar{H}3$ vectors are perpendicular for a BOX or RAW, or if the major axis vector and the minor axis vector are perpendicular for a REC. If they are not perpendicular, an error message and the pertinent vector coordinates are printed out.

The statement

```
C
C26    STORE BODY DATA AND BODY DATA POINTERS IN MASTER-ASTER ARRAY
C
C              BOX SPH RCC REC TRC ELL RAW ARB TEC TOR ARS
  300 GOTO(310,320,330,310,340,330,310,230,350,340,240),ITYPE
```

is a computed GOTO statement used to cause a transfer to the proper body section where the body dimensions and pointers will be stored in the ASTER-MASTER array.

The statements

```
C
C27    POINTER FORMAT    BOX RAW  / V  / H1 /       REC  / V  / H  /
C                                 / H2   H3 /            / R1 / R2 /
C
  310 CALL SEE3(IWH,ASTER,MASTER,FX(1),FX(2),FX(3),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=IWH*I15
      CALL SEE3(IWH,ASTER,MASTER,FX(4),FX(5),FX(6),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=MASTER(LDATA)+IWH
      CALL SEE3(IWH,ASTER,MASTER,FX(7),FX(8),FX(9),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA+1)=IWH*I15
      CALL SEE3(IWH,ASTER,MASTER,FX(10),FX(11),FX(12),
     1  LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA+1)=MASTER(LDATA+1)+IWH
      LDATA=LDATA+2
      GO TO 360
```

are used to store in the ASTER array the triplet data for the present descriptor if the body is a BOX, a RAW, or a REC, since these three bodies have the same data format (four sets of triplet data). Subroutine SEE3 is called for each set of triplet data which is to be stored or located in the ASTER array; SEE3 returns with IWH, the location (pointer) of the triplet data. These pointers are packed, two pointers per word, in the MASTER array. LDATA is then incremented for the next set of pointers to be stored in the MASTER array.

The statements

```
C
C28    POINTER FORMAT    SPH  / V  / R  /
C
  320 CALL SEE3(IWH,ASTER,MASTER,FX(1),FX(2),FX(3),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=IWH*I15
      LS1=1
      CALL SEE3(IWH,ASTER,MASTER,FX(4),FX(4),FX(4),LBOT,LDATA,NDQ,LS1)
      LS1=0
      MASTER(LDATA)=MASTER(LDATA)+IWH
      LDATA=LDATA+1
      GOTO 360
```

are used to store in the ASTER array the one set of triplet data (vertex) and the one scalar value (radius) for the present descriptor if the body is a sphere. Subroutine SEE3 is called to store or locate the triplet data (LS1 = 0) in the ASTER array; SEE3 returns with IWH, the location (pointer) of the triplet data, which is packed in the MASTER array. LS1 is set to one, and Subroutine SEE3 is again called to store or locate the scalar value (radius) in the ASTER array. SEE3 returns with IWH, the location (pointer) of the radius. LS1 is then initialized to zero, the pointer to the radius is packed into the same word as the pointer to the vertex coordinates, and LDATA is incremented for the next set of pointers to be stored in the MASTER array.

The statements

```
C
C29     POINTER FORMAT    RCC  / V  / H  /       ELL  / F1 / F2 /
C                              /    / R  /            /    / L  /
C
  330 CALL SEE3(IWH,ASTER,MASTER,FX(1),FX(2),FX(3),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=IWH*I15
      CALL SEE3(IWH,ASTER,MASTER,FX(4),FX(5),FX(6),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=MASTER(LDATA)+IWH
      LS1=1
      CALL SEE3(IWH,ASTER,MASTER,FX(7),FX(7),FX(7),LBOT,LDATA,NDQ,LS1)
      LS1=0
      MASTER(LDATA+1)=IWH
      LDATA=LDATA+2
      GO TO 360
```

are used to store in the ASTER array the two sets of triplet data and the one scalar value for the present descriptor if the body is an RCC or an ELL, since these two bodies have the same data format (two sets of triplet data followed by one scalar value). The operation of these statements is similar to that described for the sphere.

The statements

```
C
C30     POINTER FORMAT    TRC  / V  / H  /       TOR  / V  / N  /
C                              / RB / RT /            / R1 / R2 /
C
  340 CALL SEE3(IWH,ASTER,MASTER,FX(1),FX(2),FX(3),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=IWH*I15
      CALL SEE3(IWH,ASTER,MASTER,FX(4),FX(5),FX(6),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=MASTER(LDATA)+IWH
      LS1=1
      CALL SEE3(IWH,ASTER,MASTER,FX(7),FX(7),FX(7),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA+1)=IWH*I15
      CALL SEE3(IWH,ASTER,MASTER,FX(8),FX(8),FX(8),LBOT,LDATA,NDQ,LS1)
      LS1=0
      MASTER(LDATA+1)=MASTER(LDATA+1)+IWH
      LDATA=LDATA+2
```

are used to store in the ASTER array the two sets of triplet data and the two scalar values for the present descriptor if the body is a TRC or a TOR, since these two bodies have the same data format (two sets of triplet data followed by two scalar values). The operation of these statements is similar to that described for the sphere.

The statements

```
      IF(ITYPE.EQ.10)LDATA=LDATA+3
      GO TO 360
```

are used to test the present descriptor to determine if it is a torus. If it is, three additional words are reserved for later use by Subroutine TOR where the number of intersects for a given ray and the distance to the third and fourth intersects (if they occur) will be stored.

The statements

```
C
C31    POINTER FORMAT    TEC  / V   / H  /
C                             / N   / A  /
C                             / R1  / R2 /
C                             /     / RR /
C
  350 CALL SEE3(IWH,ASTER,MASTER,FX(1),FX(2),FX(3),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=IWH*I15
      CALL SEE3(IWH,ASTER,MASTER,FX(4),FX(5),FX(6),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=MASTER(LDATA)+IWH
      CALL SEE3(IWH,ASTER,MASTER,FX(7),FX(8),FX(9),LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA+1)=IWH*I15
      CALL SEE3(IWH,ASTER,MASTER,FX(10),FX(11),FX(12),
     1  LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA+1)=MASTER(LDATA+1)+IWH
      LS1=1
      CALL SEE3(IWH,ASTER,MASTER,FX(13),FX(13),FX(13),
     1  LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA+2)=IWH*I15
      CALL SEE3(IWH,ASTER,MASTER,FX(14),FX(14),FX(14),
     1  LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA+2)=MASTER(LDATA+2)+IWH
      CALL SEE3(IWH,ASTER,MASTER,FX(15),FX(15),FX(15),
     1  LBOT,LDATA,NDQ,LS1)
      LS1=0
      MASTER(LDATA+3)=IWH
      LDATA=LDATA+4
```

are used to store in the ASTER array four sets of triplet data followed by three scalar values for the present descriptor if the body is a TEC. The operation of these statements is similar to that described for the sphere.

The statements

```
C
C32   CHECK IF ANY MORE ROOM FOR SOLID DATA
C
  360 IF(LDATA.LT.NDQ)GOTO 370
      WRITE (6,917)LDATA,LBOT,NDQ
      STOP
  370 CONTINUE
```

are used to determine if the LDATA pointer is referencing a location less than the NDQ (10,000) limit. If the location is less than NDQ, the program loops to process the next body; if it is equal to or greater than NDQ, an error message is printed out with pertinent program information. The program is then terminated by the STOP statement.

The statements

```
      WRITE (6,918)
      WRITE(6,947)ITY,NBOD
      WRITE (6,945)LBASE,LRPPD,LABUT,LBODY,LBOD,LDATA,LBOT,LSCAL,LTRIP,N
     ADQ
```

are used to output three messages when the DO loop has completed entering and processing all of the body data. The first output indicates that all of the body data has been processed. The second output gives the number of times each body was used in the target description. The third output gives the starting location of the major pointers used in the program.

The statements

```
C
C33   TRANSFER ASTER(LBOT TO NDQ) TO ASTER(LDATA TO LDATA+LSUB)
C
      LD=LDATA-1
      LSUB=LBOT-LD-1
```

are used to set LD to the last word in the section that contains pointers and to compute the number of unused words between the last pointer word and the first body data word at LBOT.

The statements

```
      DO 375 I=LBOT,NDQ
      ASTER(LDATA)=ASTER(I)
      LDATA=LDATA+1
  375 CONTINUE
```

consist of a DO loop which is used to shift all of the data between the location at LBOT and the end of the MASTER-ASTER array to the locations beginning at the next available location after the pointer data.

The statement

```
C
C34   UNPACK POINTER WORDS AND RECOMPUTE POINTERS TO
C     COMPENSATE FOR TRANSFER
C
      K=LBODY+3*(NRPP+NBODY)
```

is used to compute the first location of the pointer data. The last location, LD, of the pointer data was previously computed. These values will be used as the lower and upper limits, respectively, in the following DO loop.

The statements

```
      DO 390 I=K,LD
      CALL UN2(I,I1,I2)
      IF(I1.NE.0)I1=I1-LSUB
      IF(I2.NE.0)I2=I2-LSUB
      MASTER(I)=I1*I15+I2
  390 CONTINUE
```

consist of a DO loop which unpacks each word containing two pointers and recomputes the value of the pointers by subtracting LSUB, the number of positions the body data was previously transferred. The two pointers are packed and stored into the same location.

The statements

```
C
C35   REGION STORAGE
C
      WRITE (6,920)
      N=0
      J=0
      LREGD=LDATA
      LDATA=LDATA+NRMAX
      LREGL=LDATA
```

are used to output the message REGION COMBINATION DATA; initialize two counters, N and J, to zero; and assign the value of the next available location after the body data to variable LREGD. The LREGD section will contain pointers to the region data. LDATA and LREGL are assigned the value of the next location following the last word of region pointers, and will contain region data (operator and body number).

The statement

```
C
C36    ENTER REGION DATA
C
  400 READ(5,921)IR,(IA(I),IN(I),I=1,9)
```

is used to enter data on a card containing the region number, the logical operator, and the bodies that comprise the region.

The statements

```
C
C37    CHECK VALIDITY OF REGION DATA
C
       DO 410 I=1,9
       IF(IABS(IN(I)).LE.NBODY+NRPP)GOTO 410
       WRITE (6,922)IR,I
       J=J+1
  410  CONTINUE
```

consist of a DO loop which is used to test the absolute value of each body number in the region to determine if it is a valid number (not greater than the number of bodies plus the number of RPP's used to describe the geometry). If a body number is determined to be invalid, an error message is printed out along with the region number and the number of the field on the card where the error occurs. A counter for the number of invalid body numbers is incremented by one.

The statement

```
C
C38    PACK AND STORE REGION POINTER DATA
C          LREGD    / POINTER TO REGION / NUMBER OF BODIES IN REGION /
C
       IF(IR)440,420,421
```

is used to test the region number. If it is negative (-1), all regions have been entered. If the region number is zero, the present card is a continuation of the last card entered and contains more logical operators and bodies that compose the present region. If the region number is positive, the card contains data for a new region.

The statements

```
  420 WRITE (6,923)(IA(I),IN(I),I=1,9)
      GOTO 430
```

are executed if the present card input contains additional information for the present region. The additional logical operators and the bodies that comprise the region are written out. The program then branches around the section for processing a new region card.

The statements

```
 421 N=N+1
     WRITE (6,924)IR,(IA(I),IN(I),I=1,9)
     M=LREGD+N-1
     MASTER(M)=LDATA+I15
```

are executed whenever the card entered contains a new region. N, the counter for counting the number of regions, is incremented by one, and the data on the card is written out. The pointer to the region data for the new region is then computed and stored.

The statement

```
C
C39  CHECK AND CONVERT OPERATOR TO NUMERICAL VALUE
C
 430 DO 435 I=1,9
```

is used to begin a DO loop which will convert the logical operator to a number and store it and the body number in the region data section.

The statements

```
     DO 431 K=1,8
     IF(IA(I).EQ.IAA(K))GOTO 432
 431 CONTINUE
```

consist of a DO loop which is used to determine the kind of logical operator that precedes the body number.

The statements

```
     WRITE (6,925)IA(I),I
     STOP
```

are executed if no valid logical operator was found. These statements cause an error message to be printed out and the program to terminate.

The statements

```
  432 IA(I)=IAN(K)
      IF(IN(I))433,435,434
```

are executed when a valid logical operator has been found; they are used to change the logical operator to either a one or four. The body number is then tested for a zero, positive, or negative value.

The statements

```
  433 IA(I)=4+IA(I)
      IN(I)=-IN(I)
```

are executed when the body number is negative and are used to add four to the body operator so that for a negative body number, the body operator takes on a value of eight. For a positive body number, the body operator takes on a value of one or four, with one representing an OR condition as well as a (+) condition.

The statements

```
C
C40   PACK AND STORE REGION DATA      / OPERATOR / BODY NUMBER /
C
  434 MASTER(LDATA)=IA(I)*I15+IN(I)
      LDATA=LDATA+1
      MASTER(M)=MASTER(M)+1
```

are used to store the operator and body number in the region data section and to increment to the next location in the region data section for the next operator and body number. The body count for the present region is incremented by one and stored within the same word as the pointer to the present region data.

The statements

```
      IF(LDATA.LT.NDQ)GOTO 435
      WRITE (6,927)LDATA,NDQ
      STOP
  435 CONTINUE
      GOTO 400
```

are used to determine if there is more space available in the ASTER array for region data. If the end of the ASTER array has been reached, a message is printed out stating that there is no more room for region data. The program is then terminated by the STOP statement. If more storage space is available, control is transferred to continue to enter the next of the nine fields of data from the card. If all nine fields of information have already been processed, control is transferred to enter the next card.

The statements

```
C
C41    END REGION READ - TEST NUMBER OF REGIONS
C
  440 IF(N.GE.NRMAX)GOTO 441
      WRITE (6,926)IR
      STOP
```

are executed when all of the region data cards have been processed. These statements are therefore used to determine if the number of regions entered equals the number of regions in the target geometry. If the number of regions processed is less than expected, an error message is printed out, and the STOP statement terminates the program.

The statements

```
  441 IF(J.LE.0)GOTO 442
      WRITE (6,941)
      STOP
  442 WRITE (6,928)
```

are used to test the invalid body counter to determine if any invalid body numbers were encountered. If invalid body numbers were encountered, an error message is printed out, and the STOP statement terminates the program. If no invalid body numbers were encoutnered, the message FINISH READING REGION DATA is printed out.

The statements

```
C
C42    TEST FOR REGION CHECK OPTION, CHECK REGION DATA IF NOT ZERO
C       (ERROR IF ANY POINT CAN BE IN MORE THAN ONE REGION)
C
      IF(IRCHEK.EQ.NO)GOTO 500
      WRITE (6,937)
      LL=0
      MIS=0
```

are used to determine if the validity of the region data is to be checked. If no check is to be made, the program transfers to compute the entering and leaving tables. If a check is to be performed, a line is skipped and counters LL and MIS are initialized to zero.

The statements

```
C
      DO 456 I=1,NRMAX
      JJ=I+1
```

are used to begin an overall DO loop which will compare each of the operators and body numbers of one region with the operators and body numbers of all other regions. JJ is used as an index to begin comparisons with the following region.

The statement

```
      DO 455 J=JJ,NRMAX
```

is used to begin a DO loop which will step through each of the regions for comparison with the preceding region.

The statements

```
      KRI=LREGD+I-1
      CALL UN2(KRI,LOCI,NUMI)
      KRJ=LREGD+J-1
      CALL UN2(KRJ,LOCJ,NUMJ)
```

are used to compute the location of the two pointer words for the two regions to be compared; and to unpack the pointers to the two regions and the number of bodies in each region.

The statement

```
      IF(NUMI.GE.NUMJ)GOTO 450
```

is used to compare the number of bodies in each of two of the regions to be compared to determine the smaller region, since each operator and body of the smaller region will be compared with all of the operators and bodies of the larger region.

The statements

```
      IO=NUMI
      II=NUMI
      GOTO 451
```

are executed if the first region is the smaller and are used to assign the upper limits of the two following DO loops so that each item of the smaller region is tested against each item of the larger region.

The statements

```
  450 IO=NUMJ
      II=NUMI
      L=LOCI
      LOCI=LOCJ
      LOCJ=L
```

are executed if the second region is the smaller and are used to switch the pointers and assign the upper limits of the two following DO loops so that each item of the smaller region is tested against each item of the larger region.

The statements

```
C
  451 DO 453 KO=1,IO
      KLK=LOCI+KO-1
      CALL UN2(KLK,IOPO,NBO)
      DO 452 KI=1,II
      KLK=LOCJ+KI-1
      CALL UN2(KLK,IOPI,NBI)
      IF(IOPO.NE.IOPI)GOTO 452
      IF(NBO .NE.NBI )GOTO 452
      MIS=MIS+1
      GOTO 453
  452 CONTINUE
  453 CONTINUE
```

consist of an outer DO loop and a nested DO loop which are used to perform the actual comparisons of each operator and body of the two regions. To accomplish this, the locations in each of the two regions of the packed operator and body number are computed and unpacked. The operators of the bodies in the two regions are compared; if they are the same, the body numbers are compared. If the operator and body number of the two regions are identical, the MIS counter is incremented by one. Looping continues until each operator and body number of the smaller region have been compared with each operator and body number of the larger region.

The statements

```
      IF(MIS.NE.II)GOTO 454
      WRITE (6,929)J,I
      LL=LL+1
  454 MIS=0
  455 CONTINUE
```

are used to compare the count in the MIS counter with the number of bodies in the smaller region. If they are equal, there is an area in the larger region that is identical to an area in the smaller region, indicating that a given point could be in more than one region. Since this is an illegal condition, an error message with pertinent data is printed out, and the LL counter is incremented by one. If counter MIS and the number of bodies in the smaller region are unequal, there is no area in the larger region that is identical to the smaller region. Therefore, the MIS counter is initialized to zero, and control is returned to index to the next successive region.

The statements

```
      WRITE (6,930)I
  456 CONTINUE
```

are used to print out a message each time a region has been compared with all of the remaining regions. The program is then indexed to the next region.

The statements

```
      IF(LL.GT.0)STOP
      WRITE (6,937)
```

are used to determine if a given point can be in more than one region by interrogating the count in counter LL for a greater than zero condition. If a point can be more than one region, the program is terminated by the STOP statement. If the count is zero, the program continues after the WRITE statement causes one line to be skipped.

The statements

```
C
C43   PREPARE REGION LEAVE TABLE (IS=-1) AND REGION ENTER TABLE (IS=+1)
C
  500 IS=-1
      NN=NBODY+NRPP
      LENLV=LDATA
```

are used to initialize the variable IS to -1 for computing the region leaving table; +1 is used for computing the region entering table. The statements also compute NN, the total number of bodies and RPP's used to describe the geometry for use as the upper limit of a DO loop; and initialize variable LENLV to the next available location after the region operator/body number data.

The statement

```
      DO 590 MMM=1,2
```

is used to begin a two-loop DO loop which is used to compute and store the region leaving table (IS = -1) and to compute and store the region entering table (IS = +1).

The statement

```
      DO 580 I=1,NN
```

is used to begin a DO loop which will compute and store the region leaving or entering table and pointers for each body used to describe the target geometry.

The statement

```
      M=LBODY+3*(I-1)
```

is used to compute the location of the pointer word in the LBODY section where the enter/leave region pointers and the pointers to the number of regions in the table are to be packed and stored. The LBODY section requires three words per body. The first word contains the body type and the location of the body pointers. The data for the first word was computed, packed, and stored earlier in this subroutine. The second word contains the pointers to the region enter table and the region leave table. The third word contains pointers to the number of regions in the region enter table and the number of regions in the region leave table.

The statements

```
      IF(IS.GE.0)GO TO 510
      MASTER(M+1)=MASTER(M+1)+LDATA
      GO TO 520
  510 MASTER(M+1)=MASTER(M+1)+LDATA*I15
```

are used to determine if the pointer to the region leaving table (IS = -1) or the pointer to the region entering table (IS = +1) is to be stored. If the region leaving table pointer is to be stored, it is stored in the last 15 bits of the second body pointer word. If the region entering table pointer is to be stored, it is stored in the 15 bits before the last 15 bits of the second body pointer word.

The statement

```
C
  520 DO 570 J=1,NRMAX
```

is used to begin a DO loop which will compute the leaving or entering region table and the number of regions in the entering or leaving table.

The statements

```
ITEMP=LREGD+J-1
CALL UN2(ITEMP,LOC,NC)
CALL UN2(LOC,IOP,DUM)
```

are used to compute the location of the pointer and number of bodies for the present region number and to unpack the number of bodies in the region and the operator of the first body in the region.

The statement

```
DO 560 N=1,NC
```

is used to begin a DO loop which will perform the actual testing of all of the bodies in each region, compute the leaving or entering region table, and compute the number of regions in the table.

The statements

```
MM=LOC+N-1
CALL UN2(MM,IOPER,NUM)
```

are used to locate and unpack each packed operator/body number in the region.

The statement

```
IF(NUM.NE.I)GOTO 560
```

is used to compare the test body number against the body number established by the DO 580 statement. If they do not match, the program loops to consider the next test body in the present region. If no match is found in the present region, the bodies of the following regions are compared until a match is found.

The statements

```
IF(IOP.EQ.1.OR.IOP.EQ.5)GOTO 540
IF(IOPER.GT.4)GOTO 530
```

are executed when the body number established by the DO 580 statement has been located in the region data. These statements are used to determine if the logical operator of the first body in the region where the matching body number was located is an (OR) operator. If it is not an (OR) operator, a test is made to determine if the operator of the test body number is a (-).

The statement

```
      IF(IS-1)560,550,560
```

is executed when the logical operator of the first body in the region containing the test body number is not an (OR) operator and the test body number has a (+) operator. This statement tests to determine whether the entering or leaving table is being prepared. If the leaving table is being prepared, the present region cannot be a region the ray might be in. Therefore, the program loops to determine if the test body number is used elsewhere. If the entering table is being prepared, the present region can be a region the ray might be in.

The statement

```
  530 IF(IS+1)560,551,560
```

is executed when the logical operator of the first body in the region containing the test body number is not an (OR) operator and the test body number has a (-) operator. This statement tests to determine whether the entering or leaving table is being prepared. If the entering table is being prepared, the present region cannot be a region the ray might be in. Therefore, the program loops to determine if the test body number is used elsewhere. If the leaving table is being prepared, the present region can be a region the ray might be in.

The statement

```
  540 IF(IS.LT.0)GOTO 551
```

is executed when the logical operator of the first body in the region containing the present body number is an (OR) operator. This statement is used to determine if the entering table or leaving table is being prepared and to branch to increment the count of the regions in either the entering table or leaving table.

The statements

```
550 MASTER(M+2)=MASTER(M+2)+I15
    GO TO 552
551 MASTER(M+2)=MASTER(M+2)+1
```

are used to increment the count of the number of regions in the leaving table (or entering table) for the specific body being considered. The count of the number of regions in the leaving table is located in the last 15 bits of the third word of the body pointer word set. The count for the entering table is located in the 15 bits previous to the last 15 bits in the same word.

The statements

```
552 MASTER(LDATA)=J
    LDATA=LDATA+1
```

are used to store the region number in the leaving table (or entering table) and to index to the next location.

The statements

```
    IF(LDATA.LT.NDQ)GOTO 570
    WRITE (6,931)LDATA,NDQ,MMM,I
    STOP
560 CONTINUE
570 CONTINUE
580 CONTINUE
```

are used to determine if all of the storage in the MASTER-ASTER array has been used. If it has, a message is printed out along with pertinent program variables and the program is terminated by the STOP statement. If storage is still available, the program is continued.

The statements

```
      WRITE (6,938)MMM
      IS=IS+2
  590 CONTINUE
```

are executed whenever the leaving table or entering table has been completed. A message is printed out, the variable IS is set to +1 if the leaving table has just been completed, and the program returns to prepare the entering table. If the entering table has just been completed, the variable IS is set to +3, and the program continues to the next section.

The statements

```
C
C44   RESERVE SPACE FOR RIN STORAGE, ROUT STORAGE, AND
        SUBROUTINE G1 TEMPORARY STORAGE
C
C
      L1=LDATA-1
      NN=NRPP+NBODY
      LRIN=LDATA+1
      LROT=LRIN+NN
      LIO=LROT+NN
      LEGEOM=LIO+NN
      WRITE (6,932)LEGEOM
      WRITE (6,919)LREGD,LREGL,LENLV,LRIN,LROT,LIO,LEGEOM
```

are used to assign to variable L1 the last location used for the region enter/leave table; to assign to variable NN the total number of geometric shapes used to describe the target geometry; to assign to variable LRIN the beginning location of the entry intersect data for a given ray; to assign to variable LROT the beginning location of the exit intersect data for a given ray; to assign to variable LIO the beginning location of a temporary storage area reserved for use by Subroutine G1; and to assign to variable LEGEOM the last plus one location of the geometry data. One word of storage for each geometric shape used to describe the target geometry is reserved for the entry intersect data, the exit intersect data, and the Subroutine G1 temporary storage area. The total storage for the geometry data is then printed out. In addition, the beginning location of the region data pointer section, LREGD; the beginning location of the region data section, LREGL; the beginning location of the enter/leave table, LENLV; the beginning locations of the enter and exit intersect data, LRIN and LROT; the beginning location of the temporary storage section for Subroutine G1, LIO; and the last plus one location of the geometry data, LEGEOM, are all printed out in table format.

The statements

```
C
C45    TEST REGION ENTER/LEAVE TABLE PRINT OPTION
C
       IF(IENTLV.EQ.NO)RETURN
       WRITE(6,946)
       NBNR=NBODY+NRPP
```

are used to determine if the region enter and leave tables are to be printed out. If not (IENTLV = 0), program control is returned to the MAIN program. If the enter/leave tables are to be written out (IENTLV = 1), the heading ENTER-LEAVE TABLE is printed out, and the number of geometric shapes used to describe the target geometry is assigned to variable NBNR for use as the upper limit in the following DO loop.

The statements

```
C
C46    PRINT OUT REGION ENTER/LEAVE TABLES
C
       DO 600 N=1,NBNR
       LOC=LBODY+3*(N-1)
       LOC=LOC+1
       CALL UN2(LOC,LENT,LEAV)
       LOC=LOC+1
       CALL UN2(LOC,NENT,NEAV)
       J1=LENT
       J2=LENT+NENT-1
       WRITE (6,933)N,J1,J2,(MASTER(K),K=J1,J2)
       J1=LEAV
       J2=LEAV+NEAV-1
       WRITE (6,934)N,J1,J2,(MASTER(K),K=J1,J2)
  600  CONTINUE
```

consist of a DO loop which is used to print out the enter/leave tables. This is accomplished by computing the pointers to the enter/leave tables and the number of regions in each. The tables for each geometric shape used to describe the target geometry are then printed out.

The statements

```
C
C47    TEST MASTER-ASTER ARRAY PRINT OPTION
C
       IF(IPRIN.EQ.0)RETURN
       WRITE (6,935)
```

are used to determine if the entire MASTER-ASTER array to the end of the enter/leave tables is to be printed out. If not (IPRIN = 0), program control is returned to the MAIN program. If the MASTER-ASTER array is to be printed out (IPRIN = 1), the message BEGIN ARRAY OUTPUT is printed out.

The statements

```
C
C48   PRINT OUT MASTER-ASTER ARRAY TO END OF REGION ENTER/LEAVE TABLES
C
      DO 620 K=LBASE,L1,3
      IK=K
      IK2=K+2
      M=0
      DO 610 I=IK,IK2
      M=M+1
      CALL UN2(I,I1,I2)
      NO1(M)=I1
      NO2(M)=I2
      O4(M)=ASTER(I)
      NO0(M)=I
  610 CONTINUE
      WRITE (6,936)(NO0(L),NO1(L),NO2(L),O4(L),L=1,3)
  620 CONTINUE
      RETURN
      END
C
C
```

are used to write out the MASTER-ASTER array in groups of three words at a time beginning at location LBASE, the first word of the MASTER-ASTER array, and continuing through the last word of the region enter/leave tables. Each total word is written out as well as the unpacked format of the word. Therefore, some of the output will be meaningless, since not every word in the MASTER-ASTER array is packed. When the output of the MASTER-ASTER array has been completed, program control returns to the MAIN program.

Subroutine RPPIN(LAR)

Subroutine RPPIN is called by Subroutine GENI during the geometry input and processing phase of the MAGIC program if RPP data is to be entered. This subroutine enters the RPP data, computes the number of abutting RPP's, computes the location of those abutting RPP's, computes the location of the boundary coordinate for each side of each RPP in the target geometry, and eliminates redundant boundary coordinates in the RPP data. The validity of the RPP data is checked if more than one RPP is used to describe the target geometry.

The statements

```
      DIMENSION X(6)
      DIMENSION MASTER(10000)
```

are used to dimension the MASTER array for 10,000 words and to dimension a six-element array, X, for use in entering boundary data for a given RPP.

The statements

```
      COMMON ASTER(10000)
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1    LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
      COMMON/RRPP/LRPPD,LABUT
```

are used to pass information into and out of this subroutine via COMMON statements.

The statement

```
      EQUIVALENCE(MASTER,ASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statements

```
C
  910 FORMAT(6E12.5)
  920 FORMAT(I8,17X,6F12.5)
  930 FORMAT(1H0,27HERROR IN DESCRIPTION OF RPP,I5,5X,10HMIN.GE.MAX)
  940 FORMAT(1H0,27HERROR IN DESCRIPTION OF RPP,7X,I10,10X,I10)
  950 FORMAT(10X,7HSURFACE,I5,8X,2E20.6)
```

are used to format the input data, the output data, and output messages.

The statements

```
C
      IERR=0
      N=1
C
C1    LBASE - BEGINNING LOCATION OF RPP POINTERS   RESERVE 12 WORDS/RPP
C           / I / J /    I (POINTER TO LIST OF ABUTTING RPP-S)
C           /   / K /    J (NUMBER OF RPP-S THAT ABUT THIS SURFACE)
C                        K (POINTER TO BOUNDARY COORDINATE FOR SURFACE)
C
      I=LBASE+12*NRPP
C
C2    LRPPD - BEGINNING LOCATION OF RPP BOUNDARY COORDINATES
C             THAT ARE POINTED TO BY K  (LBASE + 12 * NRPP)
C
      LRPPD=I
```

are used to initialize to zero the error count of the number of errors in the geometry input; to initialize variable N, the RPP number, to the first RPP; and to set variable I and pointer LRPPD to the beginning location of the RPP data (boundary coordinates) that follows the RPP pointer data. Pointer LRPPD is saved for printout of major pointer values by Subroutine GENI.

The statements

```
C
C3     ENTER BOUNDARY COORDINATES OF RPP
C
   10 READ(5,910)(X(J),J=1,6)
      WRITE (6,920)N,(X(J),J=1,6)
```

are used to enter and write out the six RPP bounding planes. The minimum and maximum values of the RPP are entered into six-element array X from the six fields on the input card.

The statements

```
C
C4     VERIFY MINIMUM BOUNDARY COORDINATE LESS THAN CORRESPONDING
C      MAXIMUM BOUNDARY COORDINATE
C
       DO 20 J=1,6,2
       IF(X(J).LT.X(J+1))GOTO 20
       WRITE (6,930)N
       STOP
   20  CONTINUE
```

consist of a DO loop which is used to verify that the minimum value is less than the maximum value of each of the x, y, and z RPP bounding planes. If one of the minimum values is larger than the corresponding maximum value, an error message is printed out, and the program is terminated by the STOP statement.

The statement

```
C
C5     STORE BOUNDARY COORDINATES BEGINNING AT
C      LOCATION LBASE + (12.* NRPP)
C
       DO 33 J=1,6
```

is used to begin a DO loop which is used to store the boundary coordinate for each of the RPP bounding planes in the LRPPD section of the RPP data; and to store the pointer to the boundary coordinate in the LBASE section of the RPP data.

The statements

```
       II=LBASE+12*NRPP
       L=LBASE+12*(N-1)+2*(J-1)
```

are used to initialize variable II to the beginning of the boundary coordinate data section; and to set pointer L to the location in LBASE of the side and RPP being processed.

The statement

```
    30 IF(II.LT.I)GOTO 31
```

is used to determine if pointer II, which is initially set to the beginning location of the boundary coordinate data section, is less than the pointer I, which is set to the location where the next boundary coordinate is to be stored. (Pointer II is used to search through each of the boundary coordinates previously stored in the ASTER array for a value equivalent to the present boundary value entered from the card input. If an equivalent value is located, the location, II, of the value, but not the new input, is stored, thus saving storage space).

The statements

```
       ASTER(I)=X(J)
       MASTER(L+1)=I
       I=I+1
       GOTO 33
```

are executed when none of the boundary coordinates already stored is found to be equivalent to the current value from the card input. These statements store the input from the card in the ASTER array at the next location in the LRPPD section, store location pointer I, increment to the next available location, and continue with the next boundary.

The statement

```
C
C6    CHECK FOR AND ELIMINATE REDUNDANT BOUNDARY COORDINATES
C
   31 IF(X(J).EQ.ASTER(II))GOTO 32
```

is used to compare the boundary coordinate already stored with the current boundary from the card input.

The statements

```
      II=II+1
      GOTO 30
```

are executed when the indexed boundary coordinate already stored does not equal the current boundary from the card input. These statements increment to the next boundary coordinate and cause a branch to determine if all stored boundary coordinates have been compared.

The statements

```
   32 MASTER(L+1)=I
   33 CONTINUE
```

are executed when an already stored boundary coordinate is found that is equivalent to the current boundary from the card input. These statements store the pointer to the equivalent value and continue to test the next boundary coordinate input.

The statements

```
      IF(N.GE.NRPP)GOTO 40
      N=N+1
      GOTO 10
```

are used to determine if all of the RPP data cards have been processed when each of the six fields from the current RPP data card has been processed. If not, RPP number N is incremented by one, and control is returned to enter the next RPP data card.

The statements

```
C
C7     LABUT - BEGINNING LOCATION OF LIST OF ABUT RPP-S PACKED 2/WORD
C            I POINTS HERE
C            J CONTAINS NUMBER IN LIST
C
   40 LABUT=I
      LAST=I-1
      L=LAST
```

are executed when all RPP data cards have been entered and the boundary coordinate data, with respective pointers, has been stored in the MASTER-ASTER array. These statements are used to set the pointer, LABUT, to the next location after the boundary coordinate data, which is the beginning of the packed list of abutting RPP's. Pointer LABUT is saved for printout of major pointer values by Subroutine GENI. Variables LAST and L are set to the last location of the boundary coordinate data.

The statements

```
C
C8     SEARCH FOR ABUTTING RPP-S TO SURFACE J OF RPP I
C
      DO 57 I=1,NRPP
      DO 57 N=1,6
```

are used to begin DO loops which will determine the abutting RPP's for each side of each RPP.

The statements

```
      LL=0
      M=1
```

are used to initialize to zero the number of abutting RPP's counter, LL, and to initialize switch M to +1 each time a new side of a possible abutting RPP is to be tested. M can equal either +1 or -1 and is used in determining which of two parts of a packed word the abutting RPP number is to be stored.

The statements

```
      K=LBASE+12*(I-1)+2*(N-1)
      MASTER(K)=(L+1)*I15+MASTER(K)
```

is used to compute and store the location of the list of abutting RPP's for the side of the RPP being considered.

The statement

```
      NC=3*N-1-4*(N/2)
```

is used to set variable NC equal to the opposite side number from N for the RPP that is to be tested.

The statement

```
C
C9    DETERMINE IF RPP J HAS ABUTTING SURFACE TO RPP I
C
      DO 56 J=1,NRPP
```

is used to begin a DO loop which will test each of the RPP's in the target geometry for abutting surfaces.

The statements

```
      IF(I.EQ.J)GOTO 56
      IF(S(I,N).NE.S(J,NC))GOTO 56
```

are used to insure that I and J, the two RPP's to be compared, are not the same RPP; and to cause a search for the RPP with an opposite side (NC) from side N which has an identical coordinate.

The statement

```
C
C10   COMPARE BOUNDARY COORDINATES OF RPP-S I AND J
C
      DO 53 K=1,3
```

is used to begin a DO loop which will test the opposite sides of the two RPP's other than the two sides with identical coordinates.

The statements

```
      NN=N+NC
      K41=4*K-1
      IF(NN.EQ.K41)GOTO 53
```

are used to prevent the two opposite sides with identical coordinates from being tested since it has already been determined that the sides share the same boundary.

The statements

```
      K2=2*K
      K21=K2-1
```

are used to compute two opposite faces of the two RPP's for comparing boundaries.

The statements

```
      IF(S(I,K21).GT.S(J,K21))GOTO 50
      IF(S(J,K21).LT.S(I,K2 ))GOTO 53
   50 IF(S(I,K21).GE.S(J,K2 ))GOTO 51
      IF(S(J,K2 ).LE.S(I,K2 ))GOTO 53
   51 IF(S(I,K2 ).GT.S(J,K2 ))GOTO 56
      IF(S(I,K21).LT.S(J,K21))GOTO 56
   53 CONTINUE
```

are used to compare the two opposite remaining face pairs to determine if they share the same coordinate within the boundaries of the two RPP's.

The statements

```
C
C11   STORE RPP NUMBER IN ABUTTING RPP LIST AND INCREMENT NUMBER
C
      M=-M
      IF(M.LT.0)GOTO 54
```

are used to set M to its opposite value (+1 or -1) and to test for a -1 condition to determine in which part of a two-part packed word the abutting RPP is to be stored.

The statements

```
      MASTER(L)=MASTER(L)+J
      GOTO 55
```

are executed if the next available location for storing the abutting RPP number is the last 15 bits of the storage word. The abutting RPP number is therefore stored in the last 15 bits, and control is transferred to increment the count of abutting RPP's.

The statements

```
 54 L=L+1
    MASTER(L)=J*I15
```

are executed if the next available location for storing the abutting RPP number is in the 15 bits prior to the last 15 bits of the next word. These statements therefore increment L to the next word and store the number of the abutting RPP in the inside 15 bits.

The statements

```
 55 LL=LL+1
 56 CONTINUE
```

are used to increment the count of RPP's that abut side J of RPP I and to transfer to test the next RPP.

The statements

```
    K=LBASE+12*(I-1)+2*(N-1)
    MASTER(K)=MASTER(K)+LL
 57 CONTINUE
```

are used to store the number (LL) of RPP's abutting side J of RPP I when all RPP's in the target geometry have been tested. The program then transfers to consider the next side or RPP.

The statement

```
      IF(NRPP.LE.1)GOTO 63
```

is used to test the number representing the quantity of RPP's used to describe the target geometry. If only one RPP (or none) was used, the program transfers around the section for testing the validity of RPP data.

The statement

```
C
C12   TEST VALIDITY OF RPP DATA
C
C
      DO 62 J=1,6
```

is used to begin a DO loop to verify, for each side of all RPP's, that there is either an abutting RPP; or, if the side faces the outside of the geometry, that it has the same boundary coordinate as those same sides of the other RPP's in the geometry whose sides are on the same outside boundary.

The statements

```
NRPP1=NRPP-1
DO 61 I=1,NRPP1
```

are used to set the upper limit of the DO loop to one less than the number of RPP's in the target geometry, and to begin a DO loop that will search through the RPP's for an RPP with a side that has no abutting RPP.

The statements

```
JJ=LBASE+12*(I-1)+2*(J-1)
CALL UN2(JJ,IDUM,I2)
I3=MASTER(JJ+1)
```

are used to determine the number of abutting RPP's and the pointer to the boundary coordinate for the side of the RPP being considered.

The statements

```
IF(I2.NE.0)GOTO 61
II=I+1
```

are used to test for the number of abutting RPP's. If there are RPP's abutting the side, the program loops to test the same side of the next RPP. When a side is found with no abutting RPP's, II is set to the number of the next RPP as the lower limit of the following DO loop.

The statement

```
DO 60 K=II,NRPP
```

is used to begin a DO loop which will search the remaining RPP's in the target geometry for the condition of no abutting RPP's on the same side.

The statements

```
KK=LBASE+12*(K-1)+2*(J-1)
CALL UN2(KK,IDUM,I5)
I6=MASTER(KK+1)
```

are used to determine the number of abutting RPP's and the pointer to the boundary coordinate for the RPP side.

The statements

```
      IF(I5.NE.0)GOTO 60
      IF(I3.EQ.I6)GOTO 60
```

are used to test the RPP side for abutting RPP's. If there are abutting RPP's, the program loops to test the same side of the following RPP's. When one is found with no abutting RPP's, the boundary pointer for the side of this RPP and the RPP pointer of the previous DO loop are compared to determine if they are identical. If they are, the program loops to test the next RPP.

The statements

```
      IERR=IERR+1
      WRITE (6,940)I,K
      WRITE (6,950)J,ASTER(I3),ASTER(I6)
```

are executed when the pointers to the boundary coordinate of the two sides of the two RPP's are not identical. These statements increment the error count and print out error messages with pertinent data.

The statements

```
   60 CONTINUE
      GOTO 62
   61 CONTINUE
   62 CONTINUE
```

are used to continue the search for RPP sides with no abutting RPP's to verify the exterior boundaries of the target geometry and to verify that no holes exist in the RPP data.

The statements

```
   63 LAR=L
      RETURN
      END
C
C
```

are used to return control to Subroutine GENI along with the location of the last word of RPP data when this subroutine has completed its function.

Subroutine ALBERT(FX,LBOT,NDQ,LS1)

Subroutine ALBERT is called by Subroutine GENI during the geometry input and processing phase of the MAGIC program whenever input data for an arbitrary polyhedron is to be entered. Subroutine ALBERT computes the equation of each side of six possible sides of the ARB, verifies that all four vertices of a given plane are in the same plane, determines the relative position of the side with respect to the other sides of the ARB, and stores the coefficients of the plane equation in the MASTER-ASTER array along with the location of the data for each plane.

The statements

```
      DIMENSION IA(6,4),AA(8,3),F(4),FX(6)
      DIMENSION MASTER(10000)
```

are used to dimension the MASTER array for 10,000 words and to dimension other arrays to be used in this subroutine.

The statements

```
      COMMON ASTER(10000)
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
```

are used to pass information into and out of this subroutine via COMMON statements.

The statement

```
      EQUIVALENCE(ASTER,MASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statements

```
C
  901 FORMAT(25X,6F12.5)
  902 FORMAT(10X,6(1X,4I1))
  903 FORMAT(10X,6E10.3)
  904 FORMAT(25X,6(4X,4I2))
  905 FORMAT(1H0,15HUNDEFINED PLANE)
  906 FORMAT(I5,10(E11.4))
  907 FORMAT(1H0,26HFOUR POINTS NOT IN A PLANE)
  908 FORMAT(1H0,25HERROR IN SIDE DESCRIPTION)
  909 FORMAT(1H0,16HDEGENERATE PLANE,I5)
```

are used to format input and output data and output messages.

The statements

```
C
C1    STORE COORDINATES OF FIRST TWO VERTICES IN ARRAY AA
C
      K=1
      DO 10 I=1,2
      DO 10 J=1,3
      AA(I,J)=FX(K)
      K=K+1
   10 CONTINUE
```

consist of two DO loops which are used to equate the coordinates of the first two vertices passed to this subroutine from Subroutine GENI in six-element one-dimensional array FX to the first six elements of 24-element two-dimensional array AA.

The statements

```
C
C2    ENTER COORDINATES OF REMAINING SIX VERTICES INTO ARRAY AA
C
      READ(5,903)((AA(I,J),J=1,3),I=3,8)
C
C3    ENTER ORDINAL NUMBERS OF PLANE VERTICES
C
      READ(5,902)((IA(I,J),J=1,4),I=1,6)
      WRITE  (6,901)((AA(I,J),J=1,3),I=3,8)
      WRITE  (6,904)((IA(I,J),J=1,4),I=1,6)
```

are used to enter the coordinates of the remaining six vertices into the remaining elements of array AA, to enter the four ordinal vertex numbers for each of the six faces, and to write out the coordinates of the eight vertices and the ordinal vertex numbers for six sides of the arbitrary polyhedron.

The statement

```
C
      DO 70 I=1,6
```

is used to begin a DO loop which will compute the coefficients for the equation of each of the six surfaces, verify that all of the points lie in a plane, determine the relative position of the plane with respect to the other planes, and store the data in the ASTER-MASTER array.

The statements

```
C
C4    RETRIEVE FIRST THREE VERTEX COORDINATES OF PLANE
C
      IX=IA(I,1)
      IY=IA(I,2)
      IZ=IA(I,3)
      X1=AA(IX,1)
      Y1=AA(IX,2)
      Z1=AA(IX,3)
      X2=AA(IY,1)
      Y2=AA(IY,2)
      Z2=AA(IY,3)
      X3=AA(IZ,1)
      Y3=AA(IZ,2)
      Z3=AA(IZ,3)
```

are used to assign the x, y, and z coordinates of the first three vertices of the side to individual variable names.

The statements

```
C
C5    COMPUTE COEFFICIENTS OF PLANE EQUATION
C
      D=X1*(Y2*Z3-Z2*Y3)-X2*(Y1*Z3-Z1*Y3)+X3*(Y1*Z2-Z1*Y2)
      A=(-Y2*Z3+Z2*Y3+Y1*Z3-Z1*Y3-Y1*Z2+Z1*Y2)
      B=(X2*Z3-Z2*X3-X1*Z3+X3*Z1+X1*Z2-Z1*X2)
      C=(Y2*X3-X2*Y3-Y1*X3+X1*Y3+Y1*X2-X1*Y2)
      D12=(X1-X3)**2+(Y1-Y3)**2+(Z1-Z3)**2
```

are used to compute the x, y, and z coefficients and the constant term for the equation of the plane formed by the three vertices; and to compute the square of the length between two vertices (D12).

The statements

```
C
C6     TEST FOR DEGENERATE PLANE
C
       A2B2C2=A*A+B*B+C*C
       IF(A2B2C2.NE.0.)GOTO 21
       WRITE (6,909)I
       D=ABS(D)
       GOTO 61
```

are used to compute the sum of the squares of the x, y, and z coordinates (A2B2C2) and to test the result for a value of zero. If the result is equal to zero, a message is printed out indicating that the three vertices form a degenerate plane. The absolute value of the constant term is computed, and the program transfers to store the results in the ASTER array.

The statements

```
C
C7     TEST FOR UNDEFINED PLANE
C
    21 D1210=D12*1.0E-12
       IF(A2B2C2.GT.D1210)GOTO 22
       WRITE (6,905)
       WRITE (6,906)I,A,B,C,D,D12
       IERR=IERR+1
       GOTO 70
```

are executed if it has been determined that the three vertices do not form a degenerate plane. These statements reduce the square of the length between two vertices (D12) to an extremely small value (D1210) and compare D1210 with the value of the sum of the squares of the x, y, and z coefficients (A2B2C2). If A2B2C2 is not greater than D1210, an error message indicating an undefined plane is printed out along with pertinent data about the undefined plane. The error count is incremented by one, and control is transferred to compute and test the next plane of the ARB.

The statements

```
    22 S=SQRT(A2B2C2)
       WX=A/S
       WY=B/S
       WZ=C/S
```

are used to compute the direction cosines of the perpendicular vector from the origin to the given plane.

The statements

```
C
C8    RETRIEVE COORDINATES OF FOURTH VERTEX ON PLANE
C
      IC=IA(I,4)
      X4=AA(IC,1)
      Y4=AA(IC,2)
      Z4=AA(IC,3)
```

are used to assign the x, y, and z coordinates of the fourth vertex of the plane to individual variable names.

The statements

```
C
C9    COMPUTE DISTANCE TO PLANE OF FOURTH VERTEX
C
      D2=(-D-(A*X4)-(B*Y4)-(C*Z4))/((A*WX)+(B*WY)+(C*WZ))
      D22=D2*D2
```

are used to compute the perpendicular distance from the fourth vertex to the plane formed by the first three vertices and to square the distance.

The statements

```
C
C10   DETERMINE IF FOURTH VERTEX LIES ON PLANE OF FIRST THREE VERTICES
C
      IF(D22.LE.1.01)GOTO 30
      WRITE (6,907)
      IERR=IERR+1
      WRITE (6,906)I,A,B,C,D,D12,D2
      GOTO 70
```

are used to test the squared perpendicular distance from the fourth vertex to the plane formed by the first three vertices. If greater than the allowable limit, indicating that the fourth vertex does not lie in the plane formed by the first three vertices, an error message along with pertinent data on the plane is printed out, the error count is incremented by one, and control is transferred to compute and test the next plane of the ARB.

The statements

```
C
   30 DO 31 K=1,4
      F(K)=0.
   31 CONTINUE
```

consist of a DO loop which is used to zero the four-element array, F.

The statements

```
C
C11    COMPUTE VALUES OF OTHER FOUR VERTICES
C      WITH RESPECT TO PRESENT SIDE
C
       L=1
       DO 32 J=1,8
       IF(J.EQ.IX.OR.J.EQ.IY.OR.J.EQ.IZ.OR.J.EQ.IC)GOTO 32
       F(L)=A*AA(J,1)+B*AA(J,2)+C*AA(J,3)+D
       L=L+1
    32 CONTINUE
```

are used to initialize F array index L to one. The DO loop is used to determine the other four vertex numbers that are not a part of the plane being considered. These four vertex numbers are used to retrieve the x, y, and z coordinates for each of the four vertices and substitute them into the equation for the plane being considered. The results for each are stored in four-element array F.

The statements

```
C
C12    COMPUTE NUMBER OF OTHER VERTICES ON EITHER
C      SIDE OF PLANE OR ON PLANE
C
       M=0
       N=0
       J=0
       DO 44 L=1,4
       IF(ABS(F(L)).LE.1.0E-6)GOTO 42
       IF(F(L))41,42,43
    41 M=M+1
       GOTO 44
    42 N=N+1
       GOTO 44
    43 J=J+1
    44 CONTINUE
```

are used to initialize counters M, N, and J to zero. The DO loop is used to test the results of each of the four vertices in four-element array F, to determine if the vertex lies on the plane (count in N), if the vertex lies on the positive side of the plane (count in J), or if the vertex lies on the negative side of the plane (count in M).

The statements

```
C
C13   DETERMINE SIDE OF PLANE OTHER VERTICES ARE LOCATED
C
      IF(N.EQ.0)GOTO 51
      IF(M+N.EQ.4)GOTO 60
      IF(J+N.EQ.4)GOTO 61
      GOTO 52
   51 IF(M.EQ.4)GOTO 60
      IF(J.EQ.4)GOTO 61
```

are used to determine on which side of the plane the remaining four vertices are located and to verify, by testing the contents of counters M, N, and J, that they are all on the same side of the plane.

The statements

```
   52 WRITE (6,908)
      WRITE (6,906)I,A,B,C,D,D12,D2,(F(L),L=1,4)
      IERR=IERR+1
      GOTO 70
```

are executed if the remaining four vertices not part of the plane under test are not all on the same side of the plane. These statements therefore are used to write out an error message along with pertinent program information for troubleshooting purposes. The program increments the error-count and transfers to compute and test the next side of the ARB.

The statements

```
C
   60 A=-A
      B=-B
      C=-C
      D=-D
```

are used to establish a sign convention of the ARB such that the positive side of the given plane is on the interior of the ARB. This also insures that all normals to the planes of the ARB are directed into the ARB.

The statements

```
C
C14    STORE PLANE COEFFICIENTS AND POINTERS
C
   61 CALL SEE3(IWH,ASTER,MASTER,A,B,C,LBOT,LDATA,NDQ,LS1)
      MASTER(LDATA)=IWH
      LS1=1
      CALL SEE3(IWH,ASTER,MASTER,D,D,D,LBOT,LDATA,NDQ,LS1)
      LS1=0
      MASTER(LDATA)=MASTER(LDATA)+IWH*I15
      LDATA=LDATA+1
```

are used to store in the ASTER array the coefficients A, B, and C as triplet data and the constant term D as a scalar value. Subroutine SEE3, which is called to store the triplet data (LS1=0) A, B, and C in the ASTER array, returns with location (pointer) IWH of the coefficients; this is packed in the MASTER array. LS1 is set to 1, and Subroutine SEE3 is again called to store the D term as a scalar value in the ASTER array. Subroutine SEE3 returns with location (pointer) IWH. LS1 is initialized to zero, the pointer to the D term in the ASTER array is packed into the same word as the pointer to the location of the A, B, and C terms, and LDATA is incremented for the next set of pointers for the next side of the ARB.

The statements

```
   70 CONTINUE
      RETURN
      END
C
C
```

are used to cause the subroutine to loop to compute and test the next side of the ARB. If all of the six possible sides have been computed, tested, and stored, the program returns to the calling program, Subroutine GENI.

Subroutine ARIN(LBOT,LDATA)

Subroutine ARIN is called by Subroutine GENI during the input processing phase of the MAGIC program to enter, check, process, and store the data elements of a given arbitrary surface (ARS) in the MASTER-ASTER array.

The statements

```
      SUBROUTINE ARIN(LBOT,LDATA)
C
C          SUBROUTINE READS, CHECKS, PROCESSES, AND STORES INPUT DATA
C          FOR THE ARS (ARBITRARY SURFACE)
C
      DIMENSION W(3),UW(3),VW(3),WN(3)
      DIMENSION MASTER(10000)
      COMMON ASTER(10000)
      COMMON/UNCLE/NN,IC(4)
      EQUIVALENCE (MASTER,ASTER)
```

are used to dimension arrays to be used in this subroutine, to pass information into and out of this subroutine, and to set the MASTER array equivalent to the ASTER array.

The statements

```
C
  901 FORMAT(1H ,I8,1X,3A1,2X,3HARS,2X,A4,2X,8X,
     1               37HNUMBER OF CURVES                      M=,I10 /
     2         1H ,33X,37HNUMBER OF POINTS PER CURVE          N=,I10 /
     3         1H ,33X,37HNUMBER OF POINTS IN                MN=,I10 /
     4         1H ,33X,37HNUMBER OF POINTS STORED   NP=2N(M-1)=,I10 /
     5         1H ,33X,37HTOTAL STORAGE             NSTR=4NP+82=,I10 )
  903 FORMAT(25X,6F12.4)
  904 FORMAT(10X,6F10.5)
  905 FORMAT(1H ,33X,34HNUMBER OF TRIANGLES DESCRIBED      ,I10 )
  906 FORMAT(1H ,33X,34HNUMBER OF NON-DEGENERATE TRIANGLES,I10 )
  910 FORMAT(1H0,43HERROR IN DESCRIPTION OF ARS    SOLID NUMBER,I5)
  911 FORMAT(5X,21HNUMBER OF POINTS IS 0)
  920 FORMAT(10X,2I10)
```

are used to format the input and output data entered and printed out during this subroutine.

The statements

```
C
C         ENTER NUMBER OF CURVES AND NUMBER OF POINTS PER CURVE AND
C         COMPUTE NUMBER OF POINT TO BE STORED AND STORAGE REQUIREMENTS
C
      READ(5,920)M,N
      MN=M*N
      NP=2*N*(M-1)
      NSTR=4*NP+82
      WRITE(6,901)NN,IC(1),IC(2),IC(3),IC(4),M,N,MN,NP,NSTR
```

are used to enter the number of curves which are to be input and the number of points to be input for each curve. The total number of points to be entered and the number of points to be stored are also computed. (Points are stored in pairs between consecutive curves.) The total storage required for the given ARS is computed and a message is written out giving the sequential body number of the ARS, a control number, the number of curves, the number of points per curve, the total number of points entered, the reserved space for storing the points, and the total number of storage words required for the given ARS.

The statements

```
C
C         CHECK IF NUMBER OF POINTS IS 0
C
      IF(NP.GT.0)GOTO 10
      WRITE(6,910)NN
      WRITE(6,911)
      RETURN
```

are used to determine if there are any points to be stored for the given ARS. If not, the body number and an error message are printed out, and the subroutine returns control to Subroutine GENI.

The statements

```
C
C         RESERVE STORAGE IN MASTER-ASTER ARRAY FOR ARS DATA
C
   10 LBOT=LBOT-NSTR
      MASTER(LDATA)=LBOT
      LDATA=LDATA+1
      LOC=LBOT+82
```

are used to provide a storage area in the body data section of the ASTER array for the ARS data; to store this beginning location in the body data pointer section in the MASTER array indexed by LDATA; to increment LDATA to the next available location in the body data pointer section of the MASTER array; and to set LOC to the beginning location of the ARS point data.

The statements

```
C
C         ENTER AND STORE COORDINATE DATA OF ARS
C
      LOCC=LOC+4
      DO 230 I=1,M
      IF(I.EQ.M)LOC=LOCC
      L1=LOC
      L2=LOC+8*(N-1)
      READ(5,904)(ASTER(L),ASTER(L+1),ASTER(L+2),L=L1,L2,8)
      WRITE(6,903)(ASTER(L),ASTER(L+1),ASTER(L+2),L=L1,L2,8)
      IF(I.NE.M)WRITE(6,903)
      IF(I.EQ.1.OR.I.EQ.M)GOTO 220
      DO 210 L=L1,L2,8
      ASTER(LOCC)=ASTER(L)
      ASTER(LOCC+1)=ASTER(L+1)
      ASTER(LOCC+2)=ASTER(L+2)
      LOCC=LOCC+8
  210 CONTINUE
  220 LOC=L2+8
  230 CONTINUE
```

are used to enter the M*N points and store them in the NP=2*N*(M-1) reserved locations beginning at word 82 of the reserved section for the given ARS. The points are, in effect, stored in pairs between consecutive curves. For example, given five curves (M=5) and four points per curve (N=4), the total points to be entered would be M*N=5*4=20, but would require NP=2*N*(M-1) =2*4*(5-1)=32 points of storage (four words per point). These points would be stored in the following manner (a given point is P_{MN}): P_{11}, P_{21}; P_{12}, P_{22}; P_{13}, P_{23}; $\cdots$ P_{44}, P_{54}.

The statements

```
C
C         STORE NUMBER OF POINTS STORED FOR ARS AND INITIALIZE LOCATION
C         FOR STORING NUMBER OF HITS FOR SHOTLINE
C
      MASTER(LBOT)=NP
      MASTER(LBOT+1)=0
```

are used to store the number of points stored (NP=2*N*(M-1)) in the first storage word of the given ARS data section, and to initialize the second storage word to zero. This second storage word will be later utilized by Subroutine ARS to store the number of hits on the given ARS for a given shotline.

The statements

```
C
C          ELIMINATE DEGENERATE TRIANGLES FOR GIVEN ARS DATA
C
      NT=NP-2
      WRITE(6,905)NT
      L1=LBOT+82
      L2=L1+4*(NT-1)
```

are used to compute and print out the number of triangles described by the points for the given ARS. The beginning location of the point data is stored in L1 and the first point of the last triangle is stored in L2 for use as lower and upper limits in the following DO loop for eliminating degenerate triangles in the given ARS.

The statements

```
      DO 350 L=L1,L2,4
      W(1)=ASTER(L)
      W(2)=ASTER(L+1)
      W(3)=ASTER(L+2)
      UW(1)=ASTER(L+4)-W(1)
      UW(2)=ASTER(L+5)-W(2)
      UW(3)=ASTER(L+6)-W(3)
      VW(1)=ASTER(L+8)-W(1)
      VW(2)=ASTER(L+9)-W(2)
      VW(3)=ASTER(L+10)-W(3)
      CALL CROSS(WN,UW,VW)
      IF(DOT(WN,WN).GT.0.0001)GOTO 350
      NT=NT-1
      ASTER(L+3)=-1.0
  350 CONTINUE
```

are used to check each consecutive three points to determine if they form a non-degenerate triangle or a degenerate triangle. This is accomplished by computing two vectors from the second and third points to the first point and computing the cross product of these two vectors. If the dot product of the resultant vector is zero, the three points formed a degenerate triangle, and the number of described triangles in the given ARS is reduced by one, and a negative one is placed in the fourth word of the first point of the three given points. If the dot product is greater than one, the three given points form a non-degenerate triangle. The next consecutive three points are then considered.

The statements

```
C
      WRITE(6,906)NT
      WRITE(6,903)
      RETURN
      END
```

are used to print out the number of non-degenerate triangles in the given ARS, and then to skip a line in the printout. Control is then returned to Subroutine GENI.

Subroutine SEE3(IWH,ASTER,MASTER,FX,FXX,FXXX,LBOT,LDATA,NDQ,LS1)

Subroutine SEE3 is called by Subroutine GENI or ALBERT during the input processing phase of the MAGIC program. The routine accepts either triplets or scalars and places them in the ASTER array. A search is first performed through the triplet/scalar data section to determine if the triplet or scalar already appears in the ASTER array. If so, the data will not be stored again, and location IWH of the data is returned to the calling program. If the triplet or scalar does not already appear, it is added to the ASTER array, and location IWH of the data in the array is returned to the calling program. The triplet or scalar data are passed to this routine through arguments FX, FXX, and FXXX. Argument LS1 denotes whether arguments FX, FXX, and FXXX contain triplet or scalar data. If LS1 is equal to zero, triplet data is being passed. If LS1 is equal to one, scalar data is being passed.

The statement

```
      DIMENSION ASTER(10000),MASTER(10000)
```

is used to dimension the MASTER and ASTER arrays.

The statement

```
C
C1     TEST TO DETERMINE IF TRIPLET OR SCALAR DATA
C
      IF(LS1.NE.0)GOTO 50
```

is used to test argument LS1 to determine if triplet or scalar data are being passed to this subroutine. LS1 = 1 indicates scalar data. LS1 = 0 indicates triplet data.

The statements

```
C
C2     EXECUTE IF TRIPLET DATA
C
      IF(LBOT.GT.NDQ)GOTO 20
      NDQ2=NDQ-2
```

are executed if triplet data is being passed to this subroutine. If the beginning location of the triplet/scalar data is greater than the end of the ASTER array, control is passed to store the triplet data passed through the argument list. If the beginning location of the triplet/scalar data is less than the end of the ASTER array, the upper limit of the following DO loop is computed to be two locations less than the end of the ASTER array.

The statements

```
C
C3     SEARCH FOR EQUAL TRIPLET IN THE ASTER ARRAY
C
       DO 10 I=LBOT,NDQ2
       IF(ASTER(I).NE.FX)GOTO 10
       IF(ASTER(I+1).NE.FXX)GOTO 10
       IF(ASTER(I+2).NE.FXXX)GOTO 10
       IWH=I
       RETURN
    10 CONTINUE
```

consist of a DO loop which is used to search the triplet/scalar data already in the ASTER array to determine if three succeeding quantities, equal to the triplet passed through the argument list, can be located. If so, the beginning location of the triplet is equated to pointer IWH, and control is returned to the calling program.

The statements

```
C
C4     STORE TRIPLET PASSED BY ARGUMENT LIST
C
    20 ASTER(LBOT-1)=FXXX
       ASTER(LBOT-2)=FXX
       ASTER(LBOT-3)=FX
       LBOT=LBOT-3
       IWH=LBOT
       IF(LBOT.LE.LDATA)WRITE (6,30)LBOT,LDATA
       RETURN
```

are executed if an equal triplet already in the ASTER array could not be found. These statements therefore store the triplet from the argument list backwards from the beginning of the triplet/scalar data already in storage. The beginning of the triplet/scalar data is updated to include the new data. The location of the triplet data is equaled to pointer IWH to be passed back to the calling program. If the triplet/scalar data overlaps the body data pointer section, a message with the value of the pointers of the two sections is printed.

The statement

```
C
    30 FORMAT(1H0,22HMEMORY OVERLAP IN SEE3,5X,5HLBOT=,I10,
     1  5X,6HLDATA=,I10)
```

is used to format the message and data to be written out if the body data pointer section overlaps the triplet/scalar data section.

The statements

```
C
C     EXECUTE IF SCALAR QUANTITY
C     SEARCH FOR EQUAL SCALAR QUANTITY IN THE ASTER ARRAY
C
   50 DO 60 I=LBOT,NDQ
      IF(ASTER(I).NE.FX)GOTO 60
      IWH=I
      RETURN
   60 CONTINUE
```

are executed if scalar data is being passed to this subroutine. These statements consist of a DO loop which is used to search the triplet/scalar data already in the ASTER array to determine if an equal scalar value can be located. If so, the location of the equal scalar quantity is equated to pointer IWH, and control is returned to the calling program.

The statements

```
C
C     STORE SCALAR QUANTITY PASSED BY ARGUMENT LIST
C
      ASTER(LBOT-1)=FX
      LBOT=LBOT-1
      IWH=LBOT
      RETURN
```

are executed if an equal scalar quantity could not be found. These statements therefore store the scalar from the argument list backwards from the beginning of the triplet/scalar data already in storage. The location of the scalar is equated to pointer IWH to be passed back to the calling program. The beginning location of the triplet/scalar data is also updated to include the new data.

Subroutine GRID

Subroutine GRID is the main control program for tracking a ray through the geometry. It is called by the main control program of MAGIC and controls all of the input and processing for a single attack plane after reading in data and generating the attack plane at the given angle of attack. This plane is divided into a grid of square cells, with each cell acting as the point of origin of one ray. The tracing of each ray from the grid through the vehicle geometry is accomplished by Subroutine TRACK. Subroutine GRID therefore calls Subroutine TRACK once for each ray after Subroutine GRID has defined the ray.

The statements

```
      DIMENSION WP(3)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
      COMMON/GTRACK/D1,D2,KHIT,LMAX,TR(200),XBS(3),IRSTRT,IENC,
     1  ITR(200),CA,CE,SA,SE
      COMMON/CAL/NIR,SLOS,ANGLE,NTYPE,SSPACE,L,XS(3),WS(3),
     1  TRAVEL,SN,V,H,IVIH
      COMMON/WALT/LIRFO,NG1ERR
      COMMON/HOYT/VREF,HREF
      COMMON/CELL/CELSIZ
      COMMON/CONTRL/ITESTG,IRAYSK,IENTLV,IVOLUM,IWOT,ITAPE8,NO,IYES
```

dimension a three-element array for the isotropic direction cosines resulting from a call to Subroutine TROPIC and pass information into and out of this subroutine via the COMMON statements.

The statements

```
C
  901 FORMAT(8I10)
  902 FORMAT(6E12.8)
  903 FORMAT(1H0,2HNX,I5,5X,2HNY,I5,5X,7HIRSTART,I5,5X,4HIENC,I5,5X,
     1     6HNSTART,I6,5X,4HNEND,I6,5X,9HCELL SIZE,F7.2//
     2     17H DATUM LINE AT Z=,F10.3,27H WITH RESPECT TO THE ORIGIN/
     3     17H GROUND IS  AT Z=,F10.3,27H WITH RESPECT TO THE ORIGIN/
     4     17H XSHIFT IS  AT X=,F10.3,27H WITH RESPECT TO THE ORIGIN/
     5     17H YSHIFT IS  AT Y=,F10.3,27H WITH RESPECT TO THE ORIGIN/)
```

```
904 FORMAT(1H ,7HAZIMUTH,F12.5,5X,9HELEVATION,F12.5,5X,
   1     13HBACK OFF DIST,F12.5)
905 FORMAT(2E20.8,4E10.3)
907 FORMAT(1H0,I5,15H  CELLS SKIPPED)
908 FORMAT(1H0,42HOPTION SET TO COMPUTE RANDOM POINT IN CELL)
909 FORMAT(1H0,35HOPTION SET TO CHOOSE CENTER OF CELL)
```

are used to format the grid input parameters for entering and printing out and to format output messages.

The statements

```
C
C1     READ GRID INPUT PARAMETERS
C
       READ(5,901)NX,NY,IRSTRT,IENC,NG1ERR,NSTART,NEND,ICENTR
       READ (5,902)A,E,ENGTH,ZSHIFT,GROUND
       READ (5,902)XSHIFT,YSHIFT,CELSIZ
```

are used to enter the grid input parameters. Note: it is preferred that both NX and NY always be odd.

The statements

```
C
C2     INITIALIZE PARAMETERS NOT SET BY INPUT
C
       IF(IRSTRT .LE.0)IRSTRT=1
       IF(CELSIZ .LE.0.)CELSIZ=4.
       IF(NSTART.LE.0)NSTART=1
       IF(NEND.LE.NSTART)NEND=NX*NY
       IF(NG1ERR.LE.0)NG1ERR=25
```

initialize the starting region number, the cell size, the cell number in which ray tracing is to begin, the number of the last cell, and the limit of errors allowed in Subroutine G1 if these input parameters were not initialized in the previous read statements.

The statements

```
C
C3     PRINT OUT INPUT PARAMETERS
C
       WRITE (6,903)NX,NY,IRSTRT, IENC,NSTART,NEND,CELSIZ,
      1     ZSHIFT,GROUND,XSHIFT,YSHIFT
       IF(IWOT.EQ.IYES)WRITE(1,905)A,E,XSHIFT,YSHIFT,ZSHIFT,CELSIZ
       WRITE (6,904)A,E,ENGTH
```

print out the input parameters. The variable IWOT is compared with IYES (IYES is set by program MAIN). If IWOT is not equal to one, the azimuth, the elevation, the coordinates of the center of the target, and the cell size are written out on tape 1. If IWOT is equal to zero, no writing occurs on tape 1.

The statements

```
      IF(ICENTR.EQ.0)WRITE(6,908)
      IF(ICENTR.NE.0)WRITE(6,909)
```

are used to write out a message giving the origin of the ray from the cell (random or center).

The statements

```
      RADIAN=.017453292519943
      AR=A*RADIAN
      ER=E*RADIAN
```

set the variable RADIAN to equal one degree expressed in radians and convert the azimuth and elevation angles to radians.

The statements

```
      SA=SIN(AR)
      CA=COS(AR)
      SE=SIN(ER)
      CE=COS(ER)
```

are used to compute the sine and cosine of the azimuth and elevation angles.

The statements

```
C
C       PROCESS NEND-NSTART+1 CELLS
C
        KK=NSTART
```

begin a loop to process each ray in the grid plane beginning with the first cell where the ray tracing is to begin (usually cell number one) and ending with the last cell in the grid plane (usually NX·NY).

The statements

```
C
C4      COMPUTE DIRECTION COSINES OF RAY
C
    4 WB(1)=-CE*CA
      WB(2)=-CE*SA
      WB(3)=-SE
```

compute the direction cosines for a ray perpendicular to the grid plane and directed toward the target geometry.

The statements

```
C
C5     COMPUTE ROW AND COLUMN NUMBER OF GRID CELL
C
       II=((KK-1)/NX)+1
       J=KK-(II-1)*NX
```

are used to compute the row number (II) and the column number (J) of a specific grid square from which a ray is to be started toward the target geometry.

The statements

```
C
       CELL2=.5*CELSIZ
       V=FLOAT((NY/2)-II)*CELSIZ +CELL2
       VREF=V+CELL2
       H=FLOAT((NX/2)- J)*CELSIZ +CELL2
       HREF=H+CELL2
```

are used to locate the lower left corner of the grid square. V represents the vertical distance from the center of the grid plane and H represents the horizontal distance. The variables VREF and HREF refer to the center of the specified grid square.

The statements

```
       IF(ICENTR.EQ.0)GOTO 5
       H=HREF
       V=VREF
       IVIH=0
       GOTO 6
```

are used to set the ray origin from the cell to the center of the cell if the control variable, ICENTR, is other than zero. The program then branches around the section for computing a random origin within the cell.

The statements

```
5 IV=RAN(-1)*10.
  IH=RAN(-1)*10.
  IVIH=10*IH+IV
```

are used to calculate two random numbers between zero and nine. A two-digit random number is also calculated for printout by Subroutine TRACK.

The statements

```
C
C6    COMPUTE RANDOM POINT WITHIN GRID CELL
C
      V=V+CELSIZ *FLOAT(IV)/10.+CELSIZ /20.
      H=H+CELSIZ *FLOAT(IH)/10.+CELSIZ /20.
```

locate a random point in the grid cell previously specified by variables II and J. There are 100 possible random points in the grid cell.

The statements

```
C
C7    CONVERT GRID PLANE COORDINATES TO COORDINATES OF TARGET
C
    6 XBS(1)=XSHIFT-V*CA*SE-H*SA
      XBS(2)=YSHIFT-V*SA*SE+H*CA
      XBS(3)=ZSHIFT+V*CE
```

are used to transform the point within the current grid cell into the coordinate system of the target and, at the same time, to effectively move the grid plane and target system coordinate origins to a new location specified by the variables XSHIFT, YSHIFT, and ZSHIFT.

The statement

```
      CALL TROPIC(WP)
```

calls Subroutine TROPIC to generate random direction cosines ($\overline{\text{WP}}$) from an isotropic distribution.

The statements

```
XBS(1)=XBS(1)+WP(1)*1.0E-4
XBS(2)=XBS(2)+WP(2)*1.0E-4
XBS(3)=XBS(3)+WP(3)*1.0E-4
```

are used to move the randomly selected point by a very small amount in a random direction.

The statements

```
C
CR     BACK OFF RAY ORIGIN FROM GRID PLANE TO ATTACK PLANE
C
       XB(1)=XBS(1)-ENGTH*WB(1)
       XB(2)=XBS(2)-ENGTH*WB(2)
       XB(3)=XBS(3)-ENGTH*WB(3)
```

are used to back out the origin of the ray from the target by an amount specified by the variable ENGTH. The ray from this starting point will pass through the coordinates $\overline{XBS}$ computed above.

The statement

```
IF(XB(3).LF.GROUND)GOTO 40
```

determines if the origin of the ray is below the Z coordinate of the ground. Rays will not be traced if their origin is below ground level.

The statements

```
C
CS     SAVE RAY ORIGIN AND DIRECTION COSINES OF RAY FOR LATER REFERENCE
C
       DO 20 KK1=1,3
       XS(KK1)=XB(KK1)
       WS(KK1)=WB(KK1)
    20 CONTINUE
```

consist of a DO loop which preserves the x, y, and z coordinates of the origin of ray $\overline{XB}$, and the x, y, and z coordinates of the direction cosines of ray $\overline{WB}$. Therefore, any subsequent subroutine can find the starting point and direction cosines of the ray in locations XS and SW, respectively.

The statement

```
      CALL TRACK
```

calls Subroutine TRACK to coordinate all the processing for a ray until it emerges from the far side of the vehicle and provides the calculated output for each ray. Control will not return to Subroutine GRID until calculations and outputs of results are complete for the current grid square.

The statement

```
      IF(IERR.GE.NGIERR)RETURN
```

tests to determine if the maximum allowable number of errors had occurred in Subroutine G1; if so, control is returned to the MAIN program.

The statement

```
      IF(IRAYSK.EQ.NO)GOTO 40
```

tests to determine if all grid cells are to be processed or if a random number of cells are to be skipped. The variable IRAYSK is compared to the variable NO (NO=0); if equal, no grid cells are to be skipped, and control is returned to the beginning of the DO loop to process the next cell/ray.

The statements

```
C
C10   COMPUTE RANDOM NUMBER OF CELLS (0-25) TO BE SKIPPED
C
      MSHIFT=RAN(-1)*25.
      WRITE (6,907)MSHIFT
      KK=KK+MSHIFT
```

are used to compute a random number between 0 and 24 for the random number of cells to be skipped. This value is printed out, and the index of the loop is increased by the number of cells to be skipped.

The statements

```
40 KK=KK+1
   IF(KK.LE.NEND)GOTO 4
   RETURN
   END
```

pass the control back to the beginning of the loop until the number of cells to be processed is reached. When all cells are processed, control is returned to the MAIN program.

Subroutine TRACK

Subroutine TRACK has the function of accepting grid coordinates from Subroutine GRID and initializing the "firing" of a ray. It follows the ray through all points of contact, records the pertinent data, and punches both the identification and the data cards for each ray for subsequent use in vulnerability analysis studies. This output consists of the line-of-sight thickness of each geometric region traversed by the ray, the obliquity (or angle of incidence) of the ray with respect to each intersect encountered, and the normal or perpendicular distance through the region following the intersect.

The statements

```
 DIMENSION XP(3),ERROR(2)
 COMMON/PAREM/XB(3),WB(3),IR
 COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
 COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
 COMMON/GTRACK/D1,D2,KHIT,LMAX,TR(200),XBS(3),IRSTRT,IENC,
1  ITR(200),CA,CE,SA,SE
 COMMON/CAL/NIR,SLOS,ANGLE,NTYPE,SSPACE,L,XS(3),WS(3),TRAVEL,
1  SN,V,H,IVIH
 COMMON/CONTRL/ITESTG,IRAYSK,IENTLV,IVOLUM,IWOT,ITAPE8,NO,IYES
 COMMON/WALT/LIRFO,NG1ERR
 COMMON/HOYT/VREF,HREF
 COMMON/LSU/LSURF
 COMMON/CELL/CELSIZ
 COMMON/ERR/IERR0
```

dimension a three-element array, $\overline{\text{XP}}$, which is used to store the x, y, and z coordinates of the current position of a point along the ray; dimension a two element array, ERROR, used to store alphanumeric data; and pass information into and out of this subroutine via the COMMON statements.

The statements

```
C
  901 FORMAT(F6.1,1X,F6.1,I3,1X,F7.2,1X,F7.2,4I2,I3,1X,2I3,
    1     1X,F8.3,1X,F8.3)
  902 FORMAT(2(I4,F7.2,F7.2,F6.1,I3,F7.2),1X,2I3,1X,I4,4X,A6)
  903 FORMAT(31H NUMBER OF INTERSECTIONS.GT.200)
  904 FORMAT(//)
```

```
 905 FORMAT(1H0,16H0 ITEM IN CELL (,I4,1H,,I4,1H),5X,
    1      2HH=,F6.1,5X,2HV=,F6.1)
```

are used to format data and message output.

The statements

```
C
      ERROR(2)= 6H0 ITEM
      DATA ERROR(1),ERROR(2)/4H    ,4HITEM/
```

are used to fill location ERROR(1) with four Hollerith blanks and ERROR(2) with Hollerith word ITEM.

The statements

```
      I12=4096
      NASC=-1
      IR=IRSTRT
      L=1
```

initialize program variables. I12 is set to 2^{12} or 4096 and is used for packing information into each word of the ITR array. NASC is set to -1 to indicate a new ray is being started. IR is initialized to IRSTRT, the starting region number of the ray. L represents the number of intersections and is initialized to one to start the ray since it is used as an index to store intersect data.

The statements

```
      KHIT=0
      JCNT=0
      MSKRT=0
      MTARG=1
      MARMR=0
      MVOL =0
```

initialize KHIT (the number of components hit for this cell); counter JCNT (the number of spaces encountered); skirting flag, MSKRT; armor flag, MARMR; and interior volume flag, MVOL, all to zero. The target flag, MTARG, is initialized to one.

The statements

```
C
      DO 10 I=1,200
      ITR(I)=0
      TR(I)=0.
   10 CONTINUE
```

consist of a DO loop which zeros two 200 element arrays, ITR and TR. ITR is used to store in packed form the surface number, the body number, and the number of the next region. TR is used to store the line-of-sight distance from contact to contact.

The statement

```
C
C     SUBROUTINE G1 WILL RETURN WITH S1=DISTANCE THRU REGION IR,
C     IRPRIM=THE NEXT REGION NUMBER, XP=INTERSECT OF NEXT REGION
C
   20 CALL G1(S1,IRPRIM,XP)
```

is used to call Subroutine G1. Subroutine G1 is given the current region number where the ray is located, the location of the point along the ray, and the direction cosines of the ray at that point via the COMMON statement PAREM. Subroutine G1 returns to Subroutine TRACK with the distance to the next region, the number of that region, and the new position of the point along the ray.

The statement

```
      IF(IRPRIM.LT.0)RETURN
```

tests to determine if the region number returned by Subroutine G1 is negative. If negative, an error has occurred in Subroutine G1 and control is therefore returned to Subroutine GRID.

The statement

```
      TR(L)=S1
```

stores the line-of-sight distance from the previous position of point $\overline{XP}$ to the present contact in array TR indexed by intersection counter L.

The statement

```
KLSURF=LSURF+7
```

adds seven to variable LSURF since the surface number (1-6) of the present contact could be negative if the contact is an exit intersect. This is done to avoid packing a negative number.

The statements

```
LOC=LIRFO+IR-1
CALL UN2(LOC,DUM,IDENT)
IDENT=IDENT-1
```

retrieve the space code for the region traversed for later use in this subroutine.

The statement

```
C
C2     PACK SURFACE NUMBER - BODY NUMBER - NEXT REGION NUMBER
C
       ITR(L)=(KLSURF*I12+NASC)*I12+IRPRIM
```

stores the surface number, the body number, and the number of the next region in array ITR indexed by the intersect number L. These three items are packed into one word using 12 binary bits per item.

The statements

```
IF(NASC.LE.NRPP)IRPRIM=0
IF(IRPRIM.EQ.0)GOTO 100
IR=IRPRIM
KHIT=KHIT+1
```

are used to test the body number of the present intersect against the highest RPP number since the target geometry and ray origin are contained in the highest numbered RPP; all other body numbers have higher identification numbers. If the present intersect occurs at an RPP, the next region identifier, IRPRIM, is set to zero. If IRPRIM is set to zero, the end of the ray has been reached, and control is transferred to record the results of the ray. If the present intersect is not an RPP, the present region variable is updated to the new region number in preparation for continuing the ray, and the number of components hit along the ray, KHITS, is incremented by one.

The statements

```
      IF(L.GT.1)GOTO 40
C
C3    COMPUTE DISTANCE FROM GRID PLANE TO FIRST INTERSECT OF TARGET
C
      D1=-((XP(1)-XBS(1))*WS(1) + (XP(2)-XBS(2))*WS(2)
     1    + (XP(3)-XBS(3))*WS(3))
      GOTO 60
```

are used to determine if the current intersect is the first intersect along the ray. If it is, the distance along the ray path from the intersect to the original point on the grid plane (before back-off) is computed. This distance is positive if the intersect occurs on the front side of the grid plane, or negative if the intersect occurs on the back side of the grid plane. Control is then transferred around the section that determines the space code since the intersect being considered is the first intersect.

The statements

```
C
C4    TEST SPACE IDENTIFICATION CODE
C          0 = NO SPECIAL MATERIAL  10=SKIRT  20=ARMOR  30=TARGET
C          -1,2-9,11-19,21-29,.....,91-99 = INTERIOR VOLUME
C          1 = EXTERIOR VOLUME
C
   40 IF(IDENT.EQ.0)GOTO 60
      IF(IDENT-(IDENT/10)*10.EQ.0)GOTO 50
```

are used to test the space code previously unpacked from the identification codes section in the MASTER array. If IDENT, the space code for the region just traversed, is equal to zero, no special material was in the region. If IDENT is not equal to zero, the next IF statement tests for an IDENT code of 10, 20, or 30 by utilizing integer division.

The statements

```
KHIT=KHIT-1
IF(IDENT.NE.1)MVOL=1
GOTO 60
```

are executed if IDENT is not equal to 0, 10, 20, or 30, which means that the following region is either interior volume or exterior volume. Therefore KHITS, the counter for the number of components hit along this ray, is decremented by one since the present intersect is an exit intersect into space. IDENT is compared with one to determine the type of space (1 = exterior volume). If not equal to one, the interior volume flag, MVOL, is set to one.

The statements

```
C
   50 IF(IDENT.EQ.20)MARMR=1
      IF(IDENT.EQ.30)MTARG=1
      IF(IDENT.EQ.10)MSKRT=1
```

are executed if it was previously determined that the space code, IDENT, was equal to 10, 20, or 30. These statements determine which of the three values IDENT equals and a flag is set accordingly. For IDENT = 20, the armor flag, MARMR, is set to one. For IDENT = 30, the target flag, MTARG, is set to one. For IDENT = 10, the skirt flag, MSKRT, is set to one.

The statements

```
   60 L=L+1
      IF(L.LE.200)GOTO 20
      WRITE (6,903)
      STOP
```

increment by one the index for arrays TR and ITR. If this index does not exceed 200, control is returned to locate the next intersect since there is room in storage for 200 intersections per ray. If index L becomes greater than 200, a message is written out to indicate that more than 200 intersects have occurred, and the STOP statement causes the program to be terminated.

The statements

```
C
C5     END OF RAY   OUTPUT RESULTS
C
  100 IF(L.EQ.1)RETURN
      IF(ITAPE8.EQ.NO.AND.IWOT.EQ.NO)RETURN
```

begin the section for output of the results of the ray intersections and test the variable L to determine if the present ray has had any intersections with the target. If intersections have occurred, a test is made to determine if any results are to be either printed out (ITAPE8=1) or written out on tape 1 (IWOT=1).

The statements

```
C
C6     COMPUTE DISTANCE FROM GRID PLANE TO LAST INTERSECT OF TARGET
C
      D2=XDIST(XBS,XP)-S1
      D2=-D2
```

compute the distance from the last contact on the target vehicle to the original point on the grid plane of the target (before back-off). The distance is computed by calling Function XDIST, which utilizes the standard distance formula

$$D = \sqrt{\Sigma(\overline{X}_2-\overline{X}_1)^2}$$

for determining the distance between two points. The quantity in S1 is subtracted since this is the distance between the last contact point on the target and the outer edge of the enclosing RPP (point $\overline{XP}$ on the ray is presently at the RPP intersection). The distance is set negative if the intersect occurs on the back side of the grid plane, or positive if the intersect occurs on the front side of the grid plane.

The statements

```
      IF(KHIT.GT.0)GOTO 105
      KHIT=KHIT+1
      MTARG=0
```

are used to determine if any components were hit along the ray. If no components were hit, one is added to the number of components hit variable, KHITS, since the first instruction in the next group of instructions subtracts one from KHITS. The target flag, MTARG, is set to zero since the ray missed the target.

The statement

```
  105 KHIT=KHIT-1
```

is used to decrement the number of components hit variable, KHITS, by one since the last intersect along the ray is an exit intersect of a component.

The statements

```
      IH=ABS(H/CELSIZ )+.5
      IF(H.LT.0.)IH=-IH
      IV=ABS(V/CELSIZ )+.5
      IF(V.LT.0.)IV=-IV
```

are used to compute the grid cell number in the horizontal direction (IH) and in the vertical direction (IV) from the center of the grid plane.

The statement

```
C
C7    OUTPUT GRID CELL AND TARGET IDENTIFICATION DATA
C
      IF(ITAPE8.EQ.NO)GOTO 110
```

tests to determine if output to the printer is to be suppressed (ITAPE8=0).

The statements

```
      WRITE (6,904)
      WRITE (6,901)HREF,VREF,IVIH,D1,D2,MSKRT,MTARG,MARMR,MVOL,
     1    KHIT,IH,IV,H,V
```

are executed if output to the printer is not to be suppressed (ITAPE8=1). Two lines are skipped before the cell identification data is written out.

The statement

```
 110 IF(IWOT.EQ.IYES)WRITE(1,901)HREF,VREF,IVIH,D1,D2,MSKRT,MTARG,
    1     MARMR,MVOL,KHIT,IH,IV,H,V
```

tests to determine if cell identification data is to be written out on the data output tape 1.

The statements

```
C
C*     OUTPUT RAY INTERSECTION DATA
C
       LMAX=L
       L=0
       TRAVEL=TR(1)
```

equate variable LMAX to the index of the last entry in arrays TR and ITR, set index L to 0, and store the distance travelled from the origin of the ray to the first intersect with the target in location TRAVEL.

The statement

```
       DO 200 KIK=1,LMAX,2
```

begins a DO loop for writing out the ray intersection data. Ray intersection data will be written out at two components per line. There will be less than LMAX/2 lines if spaces are encountered through the target.

The statements

```
       JERRO=1
       L=L+1
       IF(L.GE.LMAX)RETURN
```

set the index, JERRO, for the two-element array, ERROR, to a value of one, and increase index L by one to begin work on the next intersection. After incrementing L, a test is made to determine if all intersections have been processed.

The statement

```
C
C      COMPUTE DATA OUTPUT FOR FIRST HALF OF LINE
C
       CALL CALC
```

is used to call Subroutine CALC to compute information for the first half of the line of output. Subroutine CALC will compute the following each time it is called:

NIR	- Region identification (vehicle component)
SLOS	- Line-of-sight distance
ANGLE	- Obliquity angle
SN	- Normal distance through region
NTYPE	- Type of space after NIR (none=0, end ray = 9)
SSPACE	- Line-of-sight distance through space

If a space is encountered, Subroutine CALC will update index L.

The statements

```
      IF(NIR.NE.0)GOTO 113
      JERRO=2
      IERRO=IERRO+1
```

first determine if the component code is equal to zero. If it is, then the index, JERRO, for the two-element array, ERROR, is set to two, and the counter, IERRO, for the number of zero component code errors is incremented by one.

The statement

```
  113 IF(SSPACE.NE.0.)JCNT=JCNT+1
```

tests to determine if there was any space in the region. If SSPACE is not equal to zero, it contains the line-of-sight distance through space of the region. Therefore, counter JCNT, which keeps track of how many spaces are hit along the ray, is incremented by one.

The statements

```
C
C10   SAVE DATA OUTPUT FOR FIRST HALF OF LINE
C
      NIR1=NIR
      SLOS1=SLOS
      ANGLE1=ANGLE
      SN1=SN
      NTYPE1=NTYPE
      SPACE1=SSPACE
```

save the values returned from Subroutine CALC (for later output of information on the first half of the present line) in preparation for calling Subroutine CALC for data on the next intersect for output of the second half of the present line of data.

The statements

```
C
      L=L+1
      IF(L.LT.LMAX)GOTO 115
```

increment array index L by one in preparation for retrieving information about the next intersect. A test is made to determine if the last intersect has already been processed. If it has not, control is transferred to call Subroutine CALC.

The statements

```
      NIR=0
      SLOS=0.
      ANGLE=0.
      SN=0.
      NTYPE=0
      SSPACE=0.
      GOTO 120
```

zero the variables calculated by Subroutine CALC for printout on the second half of the component card if the last intersect has already been processed. Control is then transferred around the calling of Subroutine CALC.

The statement

```
C
C11   COMPUTE DATE OUTPUT FOR SECOND HALF OF LINE
C
  115 CALL CALC
```

is used to call Subroutine CALC to compute information on the next intersect for the second half of the present line of data.

The statements

```
      IF(NIR.NE.0)GOTO 117
      JERRO=2
      IERRO=IERRO+1
```

determine if the component code is equal to zero. If it is, index JERRO for two-element array ERROR is set to two, and counter IERRO, representing the number of zero component code errors is incremented by one.

The statements

```
  117 IF(SSPACE.EQ.0.)GOTO 130
  120 JCNT=JCNT+1
```

test for space in the region by determining if SSPACE has a value for line-of-sight distance through space. If space was encountered, counter JCNT, which keeps track of how many spaces are hit along the ray, is incremented by one.

The statement

```
  130 N=L-JCNT
```

computes the number of components hit by subtracting the number of spaces encountered from the number of intersects.

The statements

```
C
C12    TEST TRACK FLAG
C           501 = TRACK EDGE     502 = TRACK FACE
C           IF NORMAL DISTANCE   10 INCHES RAY ENTERS TRACK FACE
C
      IF(NIR1.NE.501)GOTO 140
      IF(SN1.LT.10.)NIR1=502
  140 IF(NIR.NE.501)GOTO 150
      IF(SN .LT.10.)NIR=502
```

are used to determine if either of the two regions for the present line of data refers to the edge of a track. If the tests result in the region being identified as the edge of a track, a test is made to determine if the normal distance for the edge of the track is less than 10 inches. If the normal distance is determined to be less than 10 inches, it is considered that the ray enters the face of the track rather than the edge, which would result in a 10-inch-or-greater normal thickness.

The statements

```
C
C13    OUTPUT RAY INTERSECTION DATA
C
  150 IF(IWOT.EQ.IYES)WRITE(1,902)NIR1,SLOS1,SN1,ANGLE1,NTYPE1,SPACE1,
     1     NIR,SLOS,SN,ANGLE,NTYPE,SSPACE,IH,IV,N
      IF(ITAPE8.EQ.IYES)WRITE(6,902)NIR1,SLOS1,SN1,ANGLE1,NTYPE1,SPACE1,
     1     NIR,SLOS,SN,ANGLE,NTYPE,SSPACE,IH,IV,N,ERROR(JERR0)
      IF(ITAPE8.EQ.NO.AND.JERR0.EQ.2)WRITE(6,905)IH,IV,HREF,VREF
```

are used to output the information computed by Subroutine TRACK and Subroutine CALC. If IWOT is equal to one, the output is written out on tape 1 and can be accessed directly by the AMSAA Armored Vehicle Vulnerability Programs. If ITAPE8 is equal to one, the output is printed out to be used for error checking. If the printer is suppressed but variable JERRO is equal to two, indicating that a zero component code error occurred, grid cell coordinate data is written out.

The statements

```
C
      IF(L.GE.LMAX)RETURN
      IF(NTYPE .EQ.9)RETURN
  200 CONTINUE
      RETURN
```

test to determine if all of the data in the TR and ITR arrays have been processed, or if NTYPE indicates that the ray has left the RPP containing the vehicle. If either condition is true, control is returned to Subroutine GRID. If neither condition is true, control is returned to the beginning of the DO loop to process the data for the next ray intersection. At the completion of the DO loop, control is returned to Subroutine GRID.

Subroutine CALC

Subroutine CALC is called by Subroutine TRACK to compute the normal distance and the angle of obliquity for each region that the ray passes through. Subroutine TRACK calls Subroutine CALC to perform these calculations after the ray has passed through all of the target geometry. Subroutine CALC also assigns a 9, indicating termination of the ray, to the space following the RPP containing the target geometry.

The statements

```
 DIMENSION XP(3),TFMP(3),TEMP1(3),TEM(3),TEM1(3),XMID(3),IEMP(4),
1  WN(3),WI(3),WA(3),XI(3),HF(3),VF(3)
 DIMENSION MASTER(10000)
 COMMON ASTER(10000)
 COMMON/PAREM/XB(3),WB(3),IR
 COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
 COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
 COMMON/GTRACK/D1,D2,KHIT,LMAX,TR(200),XBS(3),IRSTRT,IENC,
1  ITR(200),CA,CE,SA,SE
 COMMON/CAL/NIR,SLOS,ANGLE,NTYPE,SSPACE,L,XS(3),WS(3),TRAVEL,
1  SN,V,H,IVIH
 COMMON/WALT/LIRFO,NG1ERR
```

are used to dimension arrays and to pass information into and out of this subroutine.

The statement

```
 EQUIVALENCE(MASTER,ASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statement

```
 REAL NF(3)
```

is used to declare the three-element array NF(3) as a real variable.

The statements

```
C
  901 FORMAT(1H0,15HTHATS ALL FOLKS//)
  902 FORMAT(1H0,17HBAD ITYPE IN CALC,5X,6HITYPE=,I5,4HNBO=,I5/
     1   16H RETURN TO TRACK//)
  903 FORMAT(1H0,23HARS DID NOT FIND NORMAL)
  904 FORMAT(//5H NORM/5H NIR=,I10,5X,6HITYPE=,I10,5X,4HNBO=,I10,5X,
     1   6HLSURF=,I10/4H WB=,3E20.10/4H WS=,3E20.10/4H XP=,3E20.8/
     2   4H XB=,3E20.10/4H XI=,3E20.10/6H XNOS=,3E20.10)
  905 FORMAT(35H ERROR IN CALC     A TRC HAS R1 = R2 )
  906 FORMAT(42H ERROR IN CALC     BAD LSURF FOR BOX OR RAW )
```

are used to format data and message output.

The statements

```
C
C1     RETRIEVE FOR PRESENT INTERSECT THE SURFACE NUMBER,
C      BODY NUMBER, AND NEXT REGION
C
       CALL OPENK(L,LSURF,NBO,NIR)
       IF(NIR.GT.0)GOTO 10
       WRITE (6,901)
       RETURN
```

are used to first call Subroutine OPENK to unpack three items of information (surface number, body number, and the region number following the present intersect) about the present intersect, L, from the ITR array established by Subroutine TRACK. The region number following the present intersect is tested for a value of zero since zero was assigned to the region outside of the RPP containing the target geometry. If the region number is greater than zero, Subroutine CALC is continued. If not, all computations for the present ray have been completed. Therefore, a message is printed stating this fact, and control is returned to Subroutine TRACK.

The statements

```
C
C2     COMPUTE  TRAVEL - LINE-OF-SIGHT DISTANCE TO THIS REGION
C               SLOS   - LINE-OF-SIGHT DISTANCE THROUGH THIS REGION
C               XI     - COORDINATES OF INTERSECT POINT
C
   10 SLOS=TR(L+1)
```

```
      DO 20 I=1,3
      XI(I)=XS(I)+TRAVEL*WS(I)
   20 CONTINUE
      TRAVEL=TRAVEL+SLOS
      LSURF=LSURF-7
```

are used to retrieve the line-of-sight distance through the region following the present intersect. The DO loop updates the contact point coordinates from the origin of the ray to the present intersect position. The distance, TRAVEL, is updated by adding the line-of-sight distance through the region following the present intersect. Seven is subtracted from the surface number, LSURF, as unpacked by Subroutine OPENK, since seven was added before LSURF was packed by Subroutine TRACK to avoid packing a negative number.

The statements

```
C
C3    SET THE CONSTANT MULTIPLIER OF THE DIRECTION COSINES OF NORMAL
C     TO +1 FOR ENTRY OR -1 FOR EXIT
C
      XNOS=1.
      IF(LSURF.LT.0)XNOS=-1.
```

set the variable XNOS to a +1 or -1 depending upon the sign of LSURF. XNOS is a constant multiplier of the direction cosines which is used to direct the computed normal into the body if an entry intersect and away from the body if an exit intersect.

The statements

```
C
C4    RETRIEVE BODY TYPE AND LOCATION OF DATA FOR INTERSECTED BODY
C
      LOC=LBODY+3*(NBO-1)
      CALL UN2(LOC,ITYPE,LDATA)
```

are used to retrieve the body type and the location of other descriptive body data for the body where the present intersect occurs.

The statements

```
      LSURF=IABS(LSURF)
      ITYPE=ITYPE+1
      IF(ITYPE.GE.1.AND.ITYPE.LE.12)GOTO 30
      WRITE (6,902)ITYPE,NBO
      RETURN
```

are used to obtain the absolute value of the intersect surface number for use in the body routines. One is added to the variable ITYPE to avoid a zero ITYPE in the computed GOTO statements. If the variable ITYPE is equal to 1 to 12, control is passed to the correct body routine. If not, an error has occurred and an error message along with error data is printed out before returning to the calling program, Subroutine TRACK.

The statement

```
C
C5    TRANSFER TO SPECIFIC BODY SECTION TO COMPUTE DIRECTION
C     COSINES OF NORMAL
C
C          RPP BOX SPH RCC REC TRC ELL RAW ARB TEC TOR ARS
   30 GOTO(50,100,150,200,200,300,350,400,450,500,550,600),ITYPE
```

transfers control, according to the body type of the intersect being considered, to find the normal of the body.

The statements

```
C
C6    CHECK THE SPACE CODE AND ITEM CODE OF THE NEXT REGION
C
   40 CALL OPENK(L+1,DUM,DUM,NEXREG)
      ISPOT=LIRFO+NEXREG-1
      CALL UN2(ISPOT,DUM,IDENT)
      ISPOT=LIRFO+NIR-1
      CALL UN2(ISPOT,NIR,DUM)
      IDENT=IDENT-1
```

are used to retrieve the space code of the region following the next intersect, and also to retrieve the item or component code of the region following the present intersect. One is subtracted from the space code, IDENT, because a one was added when IDENT was originally packed in the MAIN program.

The statements

```
IF(IDENT-(IDENT/10)*10.NE.0)GOTO 41
NTYPE=0
SSPACE=0.
RETURN
```

test the space code of the region following the next intersect to determine if it has a special identification (10, 20, 30, · · · 80, 90). If the material has a special identification, NTYPE, the type of space following the next intersect, and SSPACE, the line-of-sight distance through space following the present intersect, are set to zero, and control is returned to Subroutine TRACK.

The statements

```
41 L=L+1
   IF(L+1.LT.LMAX)GOTO 42
```

are executed if the region following the next intersect is space. The program increments to this next intersect and tests to determine if it is the intersect with the enclosing RPP.

The statements

```
IDENT=9
SSPACE=1.0E-4
NTYPE=IDENT
RETURN
```

are executed if the program is now at the intersect with the RPP. IDENT, the space code of the space following this final intersect, and NTYPE, the type of space following this final intersect, are both set to 9 to indicate the end of the ray. The line-of-sight distance, SSPACE, through the space prior to the intersect with the RPP is set to slightly above zero so that Subroutine TRACK will count this last space as one of the spaces encountered along the ray.

The statements

```
42 NTYPE=IDENT
   SSPACE=TR(L+1)
```

```
      TRAVEL=TRAVEL+SSPACE
      RETURN
```

are executed if the next intersect is not the last intersect along the ray. The space code of the region following this next intersect is stored in NTYPE (the type of space of the region following the next intersect), the distance through the space of the region following the present intersect is retrieved from the TR array of Subroutine TRACK, and the line-of-sight distance to the next intersect is computed before returning to Subroutine TRACK.

The statements

```
C
C7    RPP SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
   50 WB(1)=0.0
      WB(2)=0.0
      WB(3)=0.0
      GOTO(51,52,53,54,55,56),LSURF
   51 WB(1)=XNOS
      GOTO 1000
   52 WB(1)=-XNOS
      GOTO 1000
   53 WB(2)=XNOS
      GOTO 1000
   54 WB(2)=-XNOS
      GOTO 1000
   55 WB(3)=XNOS
      GOTO 1000
   56 WB(3)=-XNOS
      GOTO 1000
```

are used to initialize to zero all three coordinates of the direction cosines, WB, for the normal. The coordinate for the specific direction cosine is then set depending upon which face of the RPP was intersected. The other two coordinate values of the direction cosines will remain at zero since the bounding planes are parallel to the coordinate axes.

The statements

```
C
C8    BOX SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
  100 CONTINUE
      KCOM=LSURF-(LSURF/2)*2
      IF(KCOM.EQ.0)XNOS=-XNOS
```

are used to determine if the intersected surface of the box is odd or even, and change the sign of XNOS, the constant multiplier of the direction cosines, if the surface number is even. This is done since the even numbered sides are at the ends of the length vectors, and the direction of the normal is determined from the direction of the particular length vector. The direction of the normal is directed such that the angle between the direction of the ray and the direction of the normal is less than 90 degrees. This means that the normal direction is into the body for an entry intersect and away from the body for an exit intersect.

The statements

```
    IF(LSURF-3)104,103,105
103 I=1
    GOTO 110
104 I=2
    GOTO 110
105 IF(LSURF.LT.5)GOTO 103
    I=3
```

set an index, I, according to the side number of the intersect such that I=2 for sides one or two, I=1 for sides three or four, and I=3 for sides five or six. This index is used to retrieve the coordinates from the ASTER array of the desired length vector.

The statements

```
110 CALL UN2(LDATA,IEMP(4),IEMP(1))
    LDATA=LDATA+1
    CALL UN2(LDATA,IEMP(2),IEMP(3))
```

are used to retrieve and unpack the pointers to the location of the coordinates of the vertex and length vectors in the ASTER array.

The statements

```
    DO 115 J=1,3
    LH=IEMP(I)
    LV=IEMP(4)
    M=J-1
    IJK=LH+M
    IJK1=LV+M
    TEMP(J)=ASTER(IJK)+ASTER(IJK1)
    MK=J-1+IEMP(4)
    TEMP1(J)=ASTER(MK)
115 CONTINUE
```

consist of a DO loop which retrieves from the ASTER array the coordinates of the vertex, TEMP1, and the coordinates of the end of the desired length vector, TEMP.

The statements

```
    CALL DCOSP(TEMP1,TEMP,WB)
    DO 120 J=1,3
    WA(J)=XNOS*WB(J)
120 CONTINUE
    GOTO 1000
```

compute the direction of the normal using Subroutine DCOSP. The DO loop multiplies the direction of the normal by the constant multiplier, XNOS, such that the direction of the normal is into the box for an entry intersect and away from the box for an exit intersect.

The statement

```
C
C9     SPH SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
  150 CALL UN2(LDATA,LV,DUM)
```

is used to retrieve and unpack the pointer to the location of the coordinates of the center of the sphere in the ASTER array.

The statements

```
      DO 160 I=1,3
      M=I-1+LV
      TEM(I)=ASTER(M)
  160 CONTINUE
```

consist of a DO loop which retrieves from the ASTER array the coordinates of the center of the sphere.

The statement

```
      CALL DCOSP(XI,TEM,WB)
```

is used to call Subroutine DCOSP to compute the direction cosines of the normal from the intersect point to the center of the sphere.

The statements

```
      DO 170 I=1,3
      WB(I)=XNOS*WB(I)
  170 CONTINUE
      GOTO 1000
```

are used to direct the normal such that it is into the sphere for an entry intersect and away from the sphere for an exit intersect.

The statements

```
C
C10    RCC AND REC SECTION FOR COMPUTING DIRECTION COSINES OF NORMAL
C      FOR AN INTERSECT WITH EITHER PLANAR SURFACE
C
  200 IF(LSURF-2)202,201,210
  201 XNOS=-XNOS
```

are used to determine which surface of the right circular cylinder or right elliptic cylinder has been intersected and to change the sign of XNOS, the constant multiplier of the direction cosines, if the intersected surface is the top planar surface.

The statement

```
  202 CALL UN2(LOATA,LV1,LV2)
```

begins the section for computing the direction of the normal for an intersect with either planar surface of an RCC or REC. This statement retrieves and unpacks the pointers to the location of the coordinates of the vertex and height vector in the ASTER array.

The statements

```
      DO 203 I=1,3
      M=I-1
      IJK1=M+LV1
      IJK2=M+LV2
      TEM(I)=ASTER(IJK1)
      TEM1(I)=ASTER(IJK1)+ASTER(IJK2)
  203 CONTINUE
```

consist of a DO loop which retrieves from the ASTER array the coordinates of the vertex, TEM, and the coordinates of the end of the height vector, TEM1.

The statement

```
      CALL DCOSP(TEM,TEM1,WB)
```

is used to call Subroutine DCOSP to compute the direction cosines of the height vector (the normal for an intersect with one of the planar surfaces).

The statements

```
      DO 204 I=1,3
      WB(I)=XNOS*WB(I)
  204 CONTINUE
      GOTO 1000
```

are used to direct the normal of the intersected planar surface such that its direction is into the RCC or REC for an entry intersect and away from the RCC or REC for an exit intersect.

The statements

```
C
C11    RCC AND REC SECTION FOR PROJECTING INTERSECT ONTO HEIGHT VECTOR
C      FROM THE QUADRATIC SURFACE
C
  210 CALL UN2(LDATA,LV,LH)
      LR1=MASTER(LDATA+1)
```

begin the section for computing the direction of the normal for an intersect with the quadratic surface of the RCC or REC. These statements retrieve and unpack the pointers to the location of the coordinates of the vertex and height vector and the pointer to the location of the radius in the ASTER array.

The statements

```
      DO 211 J=1,3
      M=J-1
      IJK=LV+M
      TEM(J)=ASTER(IJK)
      IJK1=LH+M
      TEM1(J)=ASTER(IJK)+ASTER(IJK1)
  211 CONTINUE
```

consist of a DO loop which retrieves from the ASTER array the coordinates of the vertex, TEM, and the coordinates of the end of the height vector, TEM1.

The statements

```
      CALL DCOSP(TEM,XI,WN)
      CALL DCOSP(TEM,TEM1,WI)
```

are used to call Subroutine DCOSP twice to compute the direction cosines of a vector from the vertex to the intersect on the quadratic surface and to compute the direction cosines of the height vector.

The statements

```
      SUM=0.
      DO 212 J=1,3
      SUM=SUM+WN(J)*WI(J)
  212 CONTINUE
```

initialize the variable SUM to zero. The DO loop computes the dot product of the two unit vectors just computed from the two calls to Subroutine DCOSP. This results in the cosines of the angle between the two vectors.

The statements

```
      DO 214 J=1,3
      XP(J)=SUM*XDIST(TEM,XI)
      XP(J)=XP(J)*WI(J)+TEM(J)
  214 CONTINUE
```

are used to compute the distance from the vertex along the height vector to the intersect projected onto the height vector, to multiply this distance by the direction codines of the height vector, and to add the coordinates of the vertex to obtain the coordinates of the intersect projected onto the height vector.

The statement

```
C
C12   TRANSFER TO REC SECTION TO COMPUTE DIRECTION COSINES OF NORMAL
C     IF AN INTERSECT ON THE QUADRATIC SURFACE OF AN REC
C
      IF(ITYPE.EQ.5)GOTO 250
```

now determines if the intersect is on an RCC or REC, since the statements from Statement 200 are common to both bodies.

The statement

```
C
C13    COMPUTE THE DIRECTION COSINES OF THE NORMAL IF AN INTERSECT ON
C      QUADRATIC SURFACE OF AN RCC
C
       CALL DCOSP(XI,XP,WB)
```

is executed if the previous test determined that the intersect is on a right circular cylinder. This statement calls Subroutine DCOSP to compute the direction cosines of the normal from the intersect point to the intersect point projected onto the height vector.

The statements

```
      DO 220 J=1,3
      WB(J)=XNOS*WB(J)
  220 CONTINUE
      GOTO 1000
```

are used to direct the normal for an intersect on the quadratic surface such that it is into the RCC for an entry intersect and away from the RCC for an exit intersect.

The statements

```
C
C14    COMPUTE THE DIRECTION COSINES OF THE NORMAL IF AN INTERSECT ON
C      QUADRATIC SURFACE OF AN REC
C
  250  LDATA=LDATA+1
       CALL UN2(LDATA,LR1,LR2)
```

are executed when it has been determined in the RCC/REC section that the intersect occurs on the quadratic surface of an REC. These statements retrieve and unpack the pointers to the coordinates of the semi-major and semi-minor axes of the REC in the ASTER array.

The statements

```
      DO 255 J=1,3
      M=J-1
      IJK1=M+LR1
      TEMP(J)=ASTER(IJK1)+XP(J)
      IJK2=M+LR2
      TEMP1(J)=ASTER(IJK2)+XP(J)
  255 CONTINUE
```

consist of a DO loop which retrieves from the ASTER array the coordinates of the semi-major axis and the semi-minor axis of the REC.

The statements

```
      A1=XDIST(XP,TEMP)
      A2=XDIST(XP,TEMP1)
```

call Subroutine XDIST twice to compute the distance between the projected intersect onto the height vector and the end of the semi-major axis; and to compute the distance between the projected intersect onto the height vector and the end of the semi-minor axis.

The statement

```
      IF(A1.GE.A2)GOTO 260
```

is used to determine if the semi-major axis of the REC is longer than or equal to the semi-minor axis.

The statements

```
      A3=A1
      A1=A2
      A2=A3
      TEMP(1)=TEMP1(1)
      TEMP(2)=TEMP1(2)
      TEMP(3)=TEMP1(3)
```

are executed if the semi-minor axis is longer than the semi-major axis. These statements exchange the values of the distance from point $\overline{XP}$ on the height vector to the coordinates of the semi-major and semi-minor axes such that the longer distance is in storage location A1. The coordinates of the semi-minor axis are then placed in location TEMP.

The statement

```
260 C=SQRT(A1*A1-A2*A2)
```

is used to compute the distance from the center of the intersected ellipse to the focus.

The statement

```
    CALL DCOSP(XP,TEMP,WN)
```

is used to call Subroutine DCOSP to compute the direction cosines of a vector from the point $\overline{XP}$ on the height vector to the farthest axis coordinate.

The statements

```
    DO 265 J=1,3
    TEM(J)=XP(J)+C*WN(J)
    TEM1(J)=XP(J)-C*WN(J)
265 CONTINUE
```

consist of a DO loop which is used to compute the coordinates of the two foci in the plane of the intersect ellipse.

The statement

```
    CALL DCOSP(TEM,XI,WN)
```

is used to call Subroutine DCOSP to compute the direction cosines of a vector from one of the foci coordinates, TEM, to the intersect point on the surface of the REC.

The statements

```
      DO 270 J=1,3
      TEM(J)=2.*A1*WN(J)+TEM(J)
  270 CONTINUE
```

consist of a DO loop which is used to compute the coordinates of the end point of a line from the focus coordinate, TEM, through the intersect point on the surface, $\overline{XI}$, with a length equal to twice the distance from the projected point on the height vector to the farthest axis point.

The statement

```
      CALL DCOSP(TEM,TEM1,WB)
```

is used to call Subroutine DCOSP to compute the direction cosines of a vector from the new point, TEM, to the other foci point, TEM1. The direction cosines of this vector are not the direction cosines of the normal of the intersect.

The statements

```
      DO 275 J=1,3
      WB(J)=XNOS*WB(J)
  275 CONTINUE
      GOTO 1000
```

are used to direct the vector for the intersect on the quadratic surface such that it is into the REC for an entry intersect and away from the REC for an exit intersect.

The statement

```
C
C15    TRC SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
  300 CALL UN2(LDATA,LV,LH)
```

is used to retrieve and unpack the pointers to the coordinates of the vertex and the height vector of the TRC in the ASTER array.

The statements

```
WN(1)=ASTER(LH)
WN(2)=ASTER(LH+1)
WN(3)=ASTER(LH+2)
```

retrieve the x, y, and z coordinates, respectively, of the height vector from the ASTER array.

The statements

```
IF(LSURF.EQ.3)GOTO 310
IF(LSURF.EQ.2)XNOS=-XNOS
CALL UNIT(WN)
WB(1)=XNOS*WN(1)
WB(2)=XNOS*WN(2)
WB(3)=XNOS*WN(3)
GOTO 1000
```

are used to test to determine if the intersect occurs on the quadratic surface of the TRC. If it does, the program branches to statement 310 to compute the normal to the quadratic surface at the intersect. If the intersect is not on the quadratic surface, the normal is computed by computing the direction cosines of the height vector and directing the result depending upon whether the top or bottom planar surface was intersected.

The statements

```
310 LDATA=LDATA+1
    CALL UN2(LDATA,LR1,LR2)
    RB=ASTER(LR1)
    RT=ASTER(LR2)
    RATIO=RB/(RB-RT)
```

are executed if the intersect occurs on the quadratic surface of the TRC and are used to retrieve the location in the ASTER array of the two radii of the TRC, and to compute the ratio of the radius of the base to the radius of the base radius minus the radius of the top.

The statements

```
      TEMP(1)=ASTER(LV)
      TEMP(2)=ASTER(LV+1)
      TEMP(3)=ASTER(LV+2)
      DO 320 I=1,3
      TEM(I)=TEMP(I)+RATIO*WN(I)-XI(I)
      TEM1(I)=TEMP(I)-XI(I)
  320 CONTINUE
```

are used to retrieve the coordinates of the vertex of the TRC, and then to compute the vector from the intersect to the apex of the TRC and the vector from the intersect to the vertex.

The statements

```
      CALL CROSS(WA,TEM,TEM1)
      CALL CROSS(WB,WA,TEM)
      CALL UNIT(WB)
```

are used to compute the cross product of the vector from the intersect to the apex of the TRC and the vector from the intersect to the vertex. The result is a vector tangent to the quadratic surface of the intersect. The cross product of the resultant vector and the vector from the intersect to the apex of the TRC results in a vector perpendicular to the quadratic surface of the intersect and directed into the TRC. The direction cosines of this vector are then computed.

The statements

```
      WB(1)=XNOS*WB(1)
      WB(2)=XNOS*WB(2)
      WB(3)=XNOS*WB(3)
      GOTO 1000
```

are used to direct the unit vector computed in the previous statement such that it is into the TRC for an entry intersect and away from the TRC for an exit intersect.

The statements

```
C
C16   ELL SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
  350 CALL UN2(LDATA,LR1,LR2)
      LS=MASTER(LDATA+1)
```

are used to retrieve and unpack the pointers to the coordinates of the two foci of the ELL in the ASTER array and to retrieve and pointer to the length of the major axis in the ASTER array.

The statements

```
      DO 352 J=1,3
      M=J-1
      IJK1=M+LR1
      IJK2=M+LR2
      TEM(J)=ASTER(IJK1)
      TEM1(J)=ASTER(IJK2)
  352 CONTINUE
```

consist of a DO loop which is used to compute the coordinates of the two foci of the ellipsoid of revolution.

The statement

```
      A=ASTER(LS)
```

retrieves the length of the major axis from the ASTER array.

The statement

```
      CALL DCOSP(TEM,XI,WN)
```

is used to call Subroutine DCOSP to compute the direction cosines of a vector from the focus, TEM, to the intersect point on the surface of the ELL.

The statements

```
      DO 353 J=1,3
      TEM(J)=A*WN(J)+TEM(J)
  353 CONTINUE
```

consist of a DO loop which is used to compute the coordinates of a point on a line from the focus, TEM, through the intersect point to a distance from focus, TEM, equal to the length of the major axis.

The statement

```
      CALL DCOSP(TEM,TEM1,WR)
```

is used to call Subroutine DCOSP to compute the direction cosines of a vector from the new computed point, TEM, to the second focus point, TEM1. The direction cosines of this vector are the direction cosines of the normal of the intersect point.

The statements

```
      DO 354 J=1,3
      WR(J)=XNOS*WH(J)
  354 CONTINUE
      GOTO 1000
```

are used to direct the normal for the intersect on the surface of the ELL such that it is into the ELL for an entry intersect and away from the ELL for an exit intersect.

The statements

```
C
C17   RAW SECTION FOR COMPUTING THE DIRECTION COSINES OF NORMAL TO
C     SLANTED SURFACE
C
  400 IF(LSURF.EQ.2)GOTO 401
```

```
C
C18   TRANSFER TO BOX SECTION IF INTERSECT NOT ON SLANT SIDE
C
      IF(LSURF.NE.4)GOTO 100
      WRITE (6*906)
      STOP
```

are used to determine if the intersect occurs on the slanted side of the right angle wedge. If it does, the program transfers to compute the normal of the slanted side. If the intersect does not occur on the slanted side and the surface number is not equal to side four (there is no side four for a RAW), the program transfers to the section for computing the normal for an intersect on a box (for sides 1, 3, 5, 6). If the surface number is equal to four, there is an error in the program since a RAW does not have a side four. Therefore, an error message is written out, and the STOP statement terminates the program.

The statements

```
401 CALL UN2(LDATA*LV*LV1)
    LDATA=LDATA+1
    CALL UN2(LDATA*LV2*LV3)
```

are used to retrieve and unpack the pointers to the coordinates of the vertex and the three length vectors if the intersect occurs on the slanted side of the RAW.

The statements

```
      DO 410 J=1*3
      M=J-1
      IJK1=M+LV1
      IJK2=M+LV2
      TEMP(J)=ASTER(IJK1)
      XMID(J)=ASTER(IJK1)-ASTER(IJK2)
      IJK3=M+LV3
      TEM(J)=ASTER(IJK3)
  410 CONTINUE
```

consist of a DO loop which is used to retrieve from the ASTER array the coordinates of the length vectors and to compute the coordinates of $\bar{H}1-\bar{H}2$.

The statements

```
      I=1
      J=2
      K=3
      LK=0
      DO 411 KK=1,3
      TEM1(I)=XMID(J)*TEM(K)-XMID(K)*TEM(J)
      LK=I
      I=J
      J=K
      K=LK
  411 CONTINUE
```

initialize integer variables for use as indices in the DO loop. The DO loop computes the coordinates of the cross product $(\overline{H}1-\overline{H}2)\times\overline{H}3$, which is a vector normal to the slanted surface, and stores the result at TEM1.

The statements

```
      SUM=0.
      DO 412 J=1,3
      SUM=TEM1(J)*TEMP(J)+SUM
  412 CONTINUE
```

initialize the variable, SUM, to zero. The DO loop computes the dot product of the vector normal to the slanted surface and the $\overline{H}1$ vector and places the result in SUM. If SUM is positive, the vector normal to the slanted side is directed from the vertex towards the slanted side. If SUM is negative, the vector is directed from the vector away from the slanted side.

The statement

```
      SUM=-SUM/ABS(SUM)
```

sets SUM equal to +1.0 if the vector normal to the slanted side is directed from the vertex away from the slanted side; or sets SUM to -1.0 if the normal is directed from the vertex towards the slanted side.

The statements

```
      TLK=TEM1(1)**2+TEM1(2)**2+TEM1(3)**2
      TLK=SQRT(TLK)
```

square the vector normal to the slanted side and compute the square root of the result which is the scalar length of the vector.

The statements

```
      DO 420 J=1,3
      WB(J)=XNOS*SUM*TEM1(J)/TLK
  420 CONTINUE
      GOTO 1000
```

are used to compute the direction cosines of the normal to the slanted side and direct it such that the normal is directed into the RAW for an entry intersect and away from the RAW for an exit intersect.

The statements

```
C
C19    ARB SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
  450 LSPT=LDATA+LSURF-1
      CALL UN2(LSPT,DUM,L1)
```

retrieve and unpack the pointer to the location in the ASTER array of the Z, Y, and Z coefficients for the equation of the intersected plane of the arbitrary polyhedron.

The statements

```
      SUM=0.
      DO 451 J=1,3
      M=J-1
      SUM=SUM+ASTER(IJK)**2
  451 CONTINUE
```

initialize storage location SUM to zero, and the DO loop computes the sum of the squares of the X, Y, and Z coefficients with the result placed in location SUM.

The statement

```
      DIV=SQRT(SUM)
```

computes the square root of the sum of the squares of the X, Y, and Z coefficients of the intersected plane of the ARB.

The statements

```
      DO 460 J=1,3
      M=J-1
      IJK=M+L1
      WB(J)=XNOS*ASTER(IJK)/DIV
  460 CONTINUE
      GOTO 1000
```

consist of a DO loop which computes the direction cosines of the normal to the intersected surface by dividing the particular coefficient by the square root of the sum of the squares of the coefficients. At the same time the normal is directed by XNOS into the ARB for an entry intersect and away from the ARB for an exit intersect.

The statements

```
C
C20   TEC SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
  500 CALL UN2(LDATA,LV,LH)
      LDATA=LDATA+1
      CALL UN2(LDATA,LN,LA)
```

are used to retrieve and unpack the pointers to the locations in the ASTER array of the coordinates of the vertex, the height vector, the normal of the base plane, and the direction of the semi-major axis of the base ellipse.

The statements

```
      WN(1)=ASTER(LN)
      WN(2)=ASTER(LN+1)
      WN(3)=ASTER(LN+2)
```

retrieve the x, y, and z coordinates of the normal to the base ellipse from the ASTER array.

The statements

```
      IF(LSURF-2)520,510,530
  510 XNOS=-XNOS
```

are used to determine which of the three surfaces of the truncated elliptic cone has been intersected and to change the sign of XNOS, the constant multiplier of the direction cosines, if the intersected surface is the top planar surface.

The statements

```
  520 WB(1)=XNOS*WN(1)
      WB(2)=XNOS*WN(2)
      WB(3)=XNOS*WN(3)
      GOTO 1000
```

are executed if the intersect occurs at either planar surface. These statements, therefore, use the normal to the base ellipse and direct it such that it is into the TEC for an entry intersect and away from the TEC for an exit intersect.

The statements

```
  530 LDATA=LDATA+1
      CALL UN2(LDATA,LR1,LR2)
      LR3=MASTER(LDATA+1)
```

begin the section for computing the normal if the intersect occurs on the quadratic surface. These statements retrieve and unpack the pointers to the location in the ASTER array of the major and minor radii of the base ellipse and retrieve the pointer to the major radius of the top ellipse.

The statements

```
      VF(1)=ASTER(LV)
      VF(2)=ASTER(LV+1)
      VF(3)=ASTER(LV+2)
      HF(1)=ASTER(LH)
      HF(2)=ASTER(LH+1)
      HF(3)=ASTER(LH+2)
```

retrieve from the ASTER array the coordinates of the vertex and height vector of the TEC.

The statements

```
      TEMP(1)=XI(1)-VF(1)
      TEMP(2)=XI(2)-VF(2)
      TEMP(3)=XI(3)-VF(3)
```

compute the vector from the vertex to the intersect point of the TEC.

The statements

```
      HH=DOT(TEMP,WN)
      HDN=DOT(HF,WN)
      GAMMA=HH/HDN
```

are used to compute the length of the vector from the vertex to the intersect point projected onto the normal to the base ellipse and to compute the length of the height vector projected onto the normal to the base ellipse (the distance between the two planar surfaces). The ratio of the height of the hit along the normal to the distance between the two planes is also computed.

The statements

```
      TEMP(1)=VF(1)+GAMMA*HF(1)
      TEMP(2)=VF(2)+GAMMA*HF(2)
      TEMP(3)=VF(3)+GAMMA*HF(3)
```

compute the coordinates of the intersect projected onto the height vector, which is the center of the intersection ellipse.

The statements

```
R1=ASTER(LR1)
R2=ASTER(LR2)
TAU=(R1/R2)**2
R4=R2/ASTER(LR3)
```

retrieve the major and minor radii of the base ellipse and compute the square of the ratio of the major radius to the minor radius of the base ellipse. The minor radius of the top ellipse is then computed.

The statements

```
BSQ=(GAMMA*R4+R2*(1.-GAMMA))**2
ASQ=TAU*BSQ
```

compute the square of the radius along the semi-minor axis of the intersection ellipse and the square of the radius along the semi-major axis of the intersection ellipse where the radius along the semi-minor axis of the intersection ellipse is given by

$$m = \gamma R4 + R2\ (1-\gamma) \tag{87}$$

and the square of the radius along the semi-major axis of the intersection ellipse is given by

$$M^2 = \tau m^2$$

The statements

```
C=SQRT(ASQ-BSQ)
TWOA=2.*SQRT(ASQ)
```

are used to compute the distance from the center of the intersect ellipse to the foci of the intersect ellipse and to compute the length of the ellipse along the semi-major axis.

The statements

```
      DO 540 I=1,3
      IJK=LA+I-1
      TEMP(I)=C*ASTER(IJK)
      TEM(I)=TEMP1(I)+TEMP(I)
      TEM1(I)=TEMP1(I)-TEMP(I)
  540 CONTINUE
```

consist of a DO loop which computes the coordinates of the foci of the intersect ellipse.

The statement

```
      CALL DCOSP(TEM,XI,WN)
```

is used to call Subroutine DCOSP to compute the direction cosines of a vector from a focus point to the intersect point on the surface of the TEC.

The statements

```
      TEMP(1)=TEM(1)+TWOA*WN(1)
      TEMP(2)=TEM(2)+TWOA*WN(2)
      TEMP(3)=TEM(3)+TWOA*WN(3)
```

are used to compute the coordinates of point along a line from a focus through the intersect at a distance equal to the length of the intersect ellipse along the semi-major axis.

The statement

```
      CALL DCOSP(TEMP,TEM1,WN)
```

is used to call Subroutine DCOSP to compute the direction cosines of a unit vector from the previously computed point to the other focus.

The statements

```
      IF(R2.EQ.R4)GOTO 545
      RATIO=R2/(R2-R4)
```

are used to compare the semi-minor radii of the base ellipse and the top ellipse of the TEC to determine if they are equal. If they are equal, the sides of the TEC are parallel and the program branches to compute the direction cosines of the normal at the intersect. If not equal, the ratio of the base semi-minor radius to the radius of the base semi-minor radius minus the semi-minor radius of the top ellipse is computed.

The statements

```
      HF(1)=VF(1)+RATIO*HF(1)-XI(1)
      HF(2)=VF(2)+RATIO*HF(2)-XI(2)
      HF(3)=VF(3)+RATIO*HF(3)-XI(3)
```

are used to compute the vector from the intersect to the apex of the TEC.

The statements

```
  545 CALL CROSS(NF,HF,WN)
      CALL CROSS(WB,NF,HF)
      CALL UNIT(WB)
```

are used to compute the cross product of the vector from the intersect to the apex of the TEC (or the $\bar{H}$ vector if the sides are parallel) and the unit vector perpendicular to the normal at the intersect. The result is a vector tangent to the quadratic surface at the intersect. The cross product of the resultant vector and the vector from the intersect to the apex of the TEC (or the $\bar{H}$ vector if the sides are parallel) is a vector perpendicular to the quadratic surface of the intersect and directed into the TEC. The direction cosines of this vector are then computer.

The statements

```
      WB(1)=XNOS*WB(1)
      WB(2)=XNOS*WB(2)
      WB(3)=XNOS*WB(3)
      GOTO 1000
```

are used to direct the unit vector for the intersect on the quadratic surface such that it is into the TEC for an entry intersect and away from the TEC for an exit intersect.

The statements

```
C
C21   TOR SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
  550 CALL UN2(LDATA,LV,LN)
      LDATA=LDATA+1
      CALL UN2(LDATA,LR1,DUM)
```

are used to retrieve and unpack the pointers to the locations in the ASTER array of the coordinates of the center of the torus, the normal to the plane of the torus, and the major radius of the torus.

The statements

```
      DO 551 I=1,3
      J=I-1
      IJK=LV+J
      TEMP(I)=XI(I)-ASTER(IJK)
      IJK=LN+J
      TEMP1(I)=ASTER(IJK)
  551 CONTINUE
```

consist of a DO loop which is used to retrieve the coordinates of the center of the torus and to compute the vector from the vertex to the intersect point on the torus. The coordinates of the normal to the plane of the torus are also retrieved.

The statement

```
      R1=ASTER(LR1)
```

is used to retrieve the value of the major radius of the torus.

The statements

```
      CALL CROSS(TEM,TEMP1,TEMP)
      CALL CROSS(TEM1,TEM,TEMP1)
      CALL UNIT(TEM1)
```

are used to compute the cross product of the normal to the plane of the torus and the vector from the center of the torus to the intersect using Subroutine CROSS. Subroutine CROSS is again called to compute the cross product of the resultant vector and the normal to the plane of the ellipse, which results in a vector from the center of the torus to the center of the circular plane of the intersect. Subroutine UNIT is then called to convert the vector to a unit vector.

The statements

```
      DO 552 I=1,3
      J=I-1
      IJK=LV+J
      TEM(I)=ASTER(IJK)
      TEMP1(I)=TEM(I)+R1*TEM1(I)
  552 CONTINUE
```

consist of a DO loop which retrieves the coordinates of the center of the torus and computes the coordinates of the center of the circular intersect plane.

The statement

```
      CALL DCOSP(TEMP1,XI,WB)
```

is used to call Subroutine DCOSP to compute the direction cosines of a vector from the intersect point to the center of the circular intersect plane, which is the normal to the intersect.

The statements

```
      DO 553 I=1,3
      WB(I)=XNOS*WB(I)
  553 CONTINUE
      GOTO 1000
```

consist of a DO loop which directs the normal for the intersect such that it is into the torus for an entry intersect and away from the torus for an exit intersect.

The statements

```
C
C22   ARS SECTION FOR COMPUTING THE DIRECTION COSINES OF THE NORMAL
C
  600 LOCARS=MASTER(LDATA)
      LOC=LOCARS+2
      DIS=XDIST(XS,XI)
```

are used to retrieve the beginning location of the ARS data in the MASTER-ASTER array and the beginning location of the ARS distance data. The distance from the origin of the ray to the present intersect is then computed.

The statements

```
      DO 610 I=1,20
      IF(ABS(DIS-ASTER(LOC)).LE.0.0001)GOTO 620
      LOC=LOC+4
  610 CONTINUE
```

consist of a DO loop used to serch the distance data for the intersect distance that is equal to the current intersect distance that was previously computed and stored.

The statements

```
      WRITE(6,903)
      SN=-1.
      ANGLE=-1.
      RETURN
```

are executed if no distance was found in the distance data that was equal to the present intersect distance. These statements, therefore, write out an error message, set the normal distance and angle to a -1 and return control to Subroutine TRACK.

The statements

```
  620 WB(1)=ASTER(LOC+1)
      WB(2)=ASTER(LOC+2)
      WB(3)=ASTER(LOC+3)
      GOTO 1000
```

are executed when a distance is found in the distance data that is equal to the current intersect distance being considered. These statements, therefore, retrieve the direction cosines of the normal to the intersected triangle that was previously stored with the intersect distance when the ray was tracked during subroutine TRACK.

The statements

```
C
C23   COMPUTE OBLIQUITY ANGLE AND NORMAL DISTANCE TO NEXT REGION
C
 1000 DO 1001 J=1,3
      XB(J)=XI(J)+WS(J)*1.0E-3
 1001 CONTINUE
```

consist of a DO loop which is used to move the coordinates of the intersect point a very small distance into the region following the intersect to avoid tracking from a boundary.

The statements

```
      ANGLE=0.
      DO 1002 J=1,3
      ANGLE=ANGLE+WB(J)*WS(J)
 1002 CONTINUE
```

initialize the variable name, ANGLE, to zero. The DO loop computes the cosine of the angle between the direction of the normal at the intersect and the direction of the ray by computing the dot product of the two unit vectors (direction cosines).

The statement

```
      IF(ABS(ANGLE).LE.1.)GOTO 1010
```

is used to determine if the absolute value of the cosine of the angle between the direction cosines of the normal and the direction cosines of the ray are less than or equal to one. A value greater than one is an error, but such an error could result from round-off error.

The statements

```
      ANGLE=0.
      SN=0.
      WRITE (6,904)NIR,ITYPE,NBO,LSURF,WB,WS,XP,XB,XI,XNOS
      IR=NIR
      GOTO 40
```

are executed if it was previously determined that the cosine of the angle between the ray and the normal is greater than one due to round-off error. The angle and normal distance are set to zero (the normal distance for a ray perpendicular to the surface at the intersect would be the same as the line-of-sight distance through the region), the values of pertinent data are written out, the region following the intersect is updated to the present region, and the program transfers to check the space code of the following region.

The statements

```
C
C24    COMPUTE OBLIQUITY ANGLE
C
 1010 ANGLE=ATAN2(SQRT(1.-ANGLE*ANGLE),ANGLE)*180./3.14159265
      IF(ANGLE.LE.90.)GOTO 1020
      DO 1011 J=1,3
      WB(J)=-WB(J)
 1011 CONTINUE
      GOTO 1000
```

convert the angle to degrees and test to determine if the angle is greater than 90 degrees due to round-off error. If it is, the direction cosines of the normal are reversed, and control is returned to again compute the angle.

The statements

```
C
C25    COMPUTE NORMAL DISTANCE TO NEXT INTERSECT
C
 1020 NASC=-2
      IR=NIR
      CALL G1(S1,IRPRIM,XP)
      SN=S1
      GOTO 40
      END
C
C
```

first sets the variable NASC to -2 to indicate to Subroutine G1 that it is to compute the normal distance through the region, not the identity of the next region. The region through which the normal passes is identified as the region following the intersect. Subroutine G1 is called to compute the normal distance through the region, which is assigned to variable SN. The program then transfers control to check the space code of the following region.

Subroutine G1(S1,IRPRIM,XP)

This subroutine is the main ray-tracing routine of the MAGIC program. It performs the following function: Given a ray in region IR at point $\overline{XB}$ with direction cosines $\overline{WB}$, determine the distance (S1) to the next region, the number (IRPRIM) of that region, the coordinates, $\overline{XP}$, of the next intersect, and the surface number (entering or leaving) of the next intersect.

Some special variables used in this subroutine are as follows:

NASC = -2	Call from CALC to find normal distance
NASC = -1	Start a new ray
IVOLUM = 1	Call from Subroutine VOLUM
ITESTG = 1	Call from Subroutine TESTG
DIST	Total distance travelled by ray

The statements

```
      DIMENSION XP(3),LSURT(50),NASCT(50)
      DIMENSION MASTER(10000)
      COMMON ASTER(10000)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
      COMMON/CAL/NIR,SLOS,ANGLE,NTYPE,SSPACE,L,XS(3),WS(3),TRAVEL,
     1  SN,V,H,IVIH
      COMMON/WALT/LIRFO,NG1ERR
      COMMON/LSU/LSURF
      COMMON/CONTRL/ITESTG,IRAYSK,IENTLV,IVOLUM,IWOT,ITAPE8,NO,IYES
      COMMON/DAVIS/IGRID,LOOP,INORM
      COMMON/CELL/CELSIZ
      COMMON/WHICH/NBO
```

dimension arrays and pass information into and out of this subroutine.

The statement

```
      EQUIVALENCE (ASTER,MASTER)
```

sets the MASTER array equivalent to the ASTER array.

The statements

```
C
  901 FORMAT(1H0,32HERROR IN G1 AT 140      BAD ITYPE,5X,4HITY=,I5)
  902 FORMAT(1H0,33HERROR IN G1 AT 510      SM %= PINF,5X,3HIR=,I5)
  903 FORMAT(4H XB=,3E20.8/4H WB=,3E20.8/10X,5HKLOOP,12X,3HNBO,
     1   12X,3HLRI,12X,3HLRO,11X,4HNHIT,11X,4HLOOP/6I15)
  904 FORMAT(1H1,15(2H* ),3X, 9HERROR NO.,I5,3X,15(2H *)//)
  905 FORMAT(34X,4HCELL,2I4)
  906 FORMAT(19H ERROR IN G1 AT 640//4H J1=,I10,4H J2=,I10,7H LSURF=,
     1   I10,6H NASC=,I10,4H IR=,I10/4H SM=,E21.10,4H S1=,E17.10/
     2   4H WB=,3E21.10/4H XB=,3E21.10)
  907 FORMAT(50H THE (SOLID POSITION/DEPTH/POINT NOW AT) IS ONE OF,
     1   6H THESE/6H XBD =,3E21.10/6H DIST=,E21.10//)
  908 FORMAT(9X,3HRIN,12X,4HROUT,7X,8HENTERING,2X,7HLEAVING,3X,
     1   8HBODY NO.,5X,3HRAY,/35X,8HSIDE NO.,2X,8HSIDE NO.//)
  910 FORMAT(//16H TILT  RIN=ROUT=,E20.10,30X,2HI=,I5//)
  911 FORMAT(2(2X,E15.8),4X,I2,8X,I2,6X,I5,5X,7HSTARTED/)
  912 FORMAT(2(2X,E15.8),4X,I2,8X,I2,6X,I5,5X,7HHAS HIT/)
  913 FORMAT(2(2X,E15.8),4X,I2,8X,I2,6X,I5,5X,7HLEAVING/)
  914 FORMAT(2(2X,E15.8),4X,I2,8X,I2,6X,I5,5X,7H IN    /)
  915 FORMAT(2(2X,E15.8),4X,I2,8X,I2,6X,I5,5X,8HENTERING/)
  916 FORMAT(2(2X,E15.8),4X,I2,8X,I2,6X,I5,5X,8HWILL HIT/)
  917 FORMAT(//4(14H END ERROR NO.,I4,3X)/)
  918 FORMAT(1H0,I5,21H ERRORS IN G1, RETURN)
```

are used to format output when errors are detected and to format a diagnostic error printout.

The statements

```
C
        INORM=0
        IF(NASC.EQ.-2)INORM=1
        S1=0.
        IF(NASC.GT.0)GOTO 20
```

initialize INORM to 0 (INORM is used in Subroutine ARS to determine whether the normal distance, INORM=1, or the line-of-sight distance, INORM=0, is to be computed). The variable NASC is tested to determine if this subroutine has been called by Subroutine CALC to compute the normal distance through a body (INORM=1). The variable S1 for recording the distance through a region is initialized to zero. If NASC is greater than zero, a branch is made to

Statement 20 to begin tracing the ray through the region. If NASC is less than zero, a new ray (either for normal or line-of-sight distance) is to be started.

The statements

```
C
C1     INITIALIZE FOR NEW RAY
C
       DIST=0.
       IF(KLOOP.LT.32000)GOTO 15
       KLOOP=0
       LION=LIO+NBODY+NRPP-1
       DO 10 I=LIO,LION
       MASTER(I)=0
    10 CONTINUE
```

are executed when a new region has been found and the ray through the region is to be started. The variable DIST for the distance through the region is initialized to zero, and the variable KLOOP is tested for less than 32,000 since the maximum size of KLOOP is 15 bits or 32,768. If KLOOP is greater than 32,000, KLOOP and the temporary data area for Subroutine G1 are initialized to zero.

The statement

```
    15 KLOOP=KLOOP+1
```

increments the count in KLOOP by one. This value is stored for each body in the region after the body routines return with the computed values of RIN and ROUT. When G1 is later called by CALC to compute the normal distance through the region, KLOOP is again incremented by one. The new value of KLOOP is compared with the previously stored value of KLOOP as a test to determine if the normal distance is to be computed. This test is performed in Subroutine G1, Subroutine WOWI, Subroutine TOR, and Subroutine ARS. Therefore, until Subroutine CALC calls for the normal calculations, the variable KLOOP will equal the previously stored value of KLOOP, now referred to as LOOP, which indicates to the above subroutines that the line-of-sight distance is to be computed.

The statements

```
C
C2     BEGIN/CONTINUE TRACING RAY THRU REGION
C
    20 SM=PINF
       NHIT=0
```

```
C
C3    COMPUTE LOCATION OF REGION DATA
C
      LOC=LREGD+IR-1
C
C4    RETRIEVE THE NUMBER OF BODIES IN REGION
C
      CALL UN2(LOC,LOC,NC)
      LOC=LOC-1
```

initialize the variable SM to an extremely large positive value. (SM is used to represent the shortest distance from the origin to the next intersect.) The number of solids hit, NHITS, is initialized to zero, and the number of descriptors in the prescribed region along with the location of the solid numbers in the region is retrieved. The location of the solid numbers is decremented by one since, in the following DO loop, one is added to the location number before the solid number is retrieved.

The statement

```
C
      DO 500 N=1,NC
```

begins a DO loop to perform the following:

Given: NC = number of bodies in region description.

Find: RIN and ROUT for each of these bodies.

RIN is the distance from point $\overline{XB}$ to the point where the ray enters the body.

ROUT is the distance from point $\overline{XB}$ to the point where the ray leaves the body.

If ROUT = -PINF, the ray does not hit the body.

Unique value of RIN indicates next body in the path of the ray.

Two or more values of RIN indicates two or more bodies have a common surface.

ROUT for current body means that the ray will leave this body before encountering another body.

G1 selects the smallest of the RIN or ROUT distance and sets it equal to DIST.

The statements

```
C
C5    RETRIEVE BODY NUMBER
C
      LOC=LOC+1
      CALL UN2(LOC,DUM,NBO)
C
C6    RETRIEVE ENTER AND EXIT SURFACE NUMBERS AND LAST RAY NUMBER
C
      ITEMP=LIO+NBO-1
      CALL UN3(ITEMP,LRI,LRO,LOOP)
C
C7    RETRIEVE BODY TYPE AND LOCATION OF DATA
C
      ITEMP=LBODY+3*(NBO-1)
      CALL UN2(ITEMP,ITYPE,LOCDA)
```

retrieve the solid number, the value of the variable LOOP for testing against KLOOP, and the type of body that the solid represents.

The statement

```
      IF(LOOP.NE.KLOOP)GOTO 130
```

determines if RIN and ROUT for the current solid have already been computed for line-of-sight distance or for the normal distance if G1 was called to compute the normal distances. If RIN and ROUT for either condition have not yet been computed, the program branches to prepare for computing RIN and ROUT (normal or line-of-sight) of the body. If RIN and ROUT have already been computed, RIN and ROUT are retrieved from storage.

The statements

```
C
C8    CONTINUE RAY   RETRIEVE RIN/ROUT FOR CURRENT BODY
C
      IF(ITYPE.GT.11)GOTO 140
      IJK=LRIN+NBO-1
      RIN=ASTER(IJK)
      IJK=LROT+NBO-1
      ROUT=ASTER(IJK)
```

are executed if RIN and ROUT were previously computed for the current body. These statements determine if the present body is a valid type and, if not, branch to record an error. If the body type is valid, the RIN and ROUT of the body are retrieved from their locations in the ASTER array.

The statements

```
      IF(ITYPE.LT.10)GOTO 320
C
C9    IS NEXT RIN/ROUT SET REQUIRED FOR TOR OR ARS
C
      IF(ROUT.LT.0.)GOTO 32r
      IF(DIST.LT.ROUT)GOTO 320
```

determine if the body type is a torus or an arbitrary surface, because these two bodies can have more than one RIN/ROUT set; if a torus or an arbitrary surface, a test is made to determine if the ray misses or if the present point along the ray is equal to or past the stored ROUT distance. If the ray misses or the distance along the ray is not yet past the ROUT, the program branches around the section for calling the body subroutine to compute the next RIN/ROUT set.

The statement

```
      IF(NASC.EQ.NBO)NASC=0
```

is executed only when the present body type is a torus or an arbitrary surface and the distance, DIST, is equal to or greater than the ROUT of the body. The statement initializes NASC to zero if it is equal to the current solid number.

The statements

```
C
  130 LRI=1
      LRO=1
      ITY=ITYPE+1
      IF(ITY.GE.1.AND.ITY.LE.12)GOTO 200
```

initialize the entering and leaving surface variables, increment the body type to avoid a zero index in the following computed GOTO statement, and test the body type number to determine if it is a valid number. If it is, the body routine is called. If not, an error printout is executed.

The statements

```
140 IERR=IERR+1
    WRITE (6,901)ITYPE
    GOTO 800
```

print out an error message if the body type number is not valid and transfer to the end of the subroutine.

The statements

```
C
C10    COMPUTE RIN/ROUT FOR CURRENT BODY
C
C             RPP BOX SPH RCC REC TRC ELL RAW ARB TEC TOR ARS
  200 GOTO(205,210,215,220,225,230,235,240,245,250,255,260),ITY
  205 CALL RPP(NBO)
      GOTO 300
  210 CALL BOX
      GOTO 300
  215 CALL SPH
      GOTO 300
  220 CALL RCC
      GOTO 300
  225 CALL REC
      GOTO 300
  230 CALL TRC
      GOTO 300
  235 CALL ELL
      GOTO 300
  240 CALL RAW
      GOTO 300
  245 CALL ARB
      GOTO 300
  250 CALL TEC
      GOTO 300
  255 CALL TOR
      GOTO 300
  260 CALL ARS
```

transfer control according to body type to a statement that calls the proper body subroutine. Upon return from the body subroutine, control is transferred to Statement 300.

The statements

```
C
C11   STORE RIN AND ROUT FOR BODY IN RIN AND ROUT TABLES
C
  300 IJK=LRIN+NBO-1
      ASTER(IJK)=RIN
      IJK=LROT+NBO-1
      ASTER(IJK)=ROUT
      IJK=LIO+NBO-1
      MASTER(IJK)=KLOOP+I15*(LRO+64*LRI)
```

store the values of RIN and ROUT in the ASTER array in the RIN and ROUT table according to body number. The entering surface number, the leaving surface number, and the value of KLOOP are also packed into a word and stored according to body number and region number.

The statements

```
C
C12   IS POINT XP ON CURRENT BODY   YES-IS IT ENTER OR EXIT
C
  320 IF(NASC.NE.NBO)GOTO 330
      IF(LSURF)500,500,340
```

compare the present body with the body where the point $\overline{XP}$ is presently located. If point $\overline{XP}$ is on the body now under test, it is determined if $\overline{XP}$ is at a RIN or ROUT location. If $\overline{XP}$ is at a ROUT location, control is passed to the end of the DO loop to consider the next body in the region. If the point $\overline{XP}$ is not on the body now under test, or is at the RIN location, the subroutine branches to perform more tests relative to the location of point $\overline{XP}$ with respect to the bodies in the region.

The statements

```
C
C13   DOES RAY INTERSECT BODY   YES-DOES IT ORIGINATE WITHIN BODY
C
  330 IF(ROUT.LE.0.)GOTO 500
      IF(RIN.GT.0.)GOTO 350
```

determine if the ray misses the body; if not, RIN is tested to determine if the present location of $\overline{XP}$ is within the body or at location RIN.

The statements

```
C
C14    POINT XP AT RIN OR WITHIN BODY
C
  340 IF(ABS(ROUT-SM).GT.SM*1.0E-6)GOTO 341
      ROUT=SM
      IJK=LROT+NBO-1
      ASTER(IJK)=ROUT
      GOTO 345
```

are executed if point $\overline{XP}$ is now at RIN on the present body or if the ray originates within the body. These statements determine if the ROUT and the distance to the last body tested are close enough to be considered a single intersect. If they are, the distance of point $\overline{XP}$ is assigned to ROUT.

The statements

```
  341 IF(ROUT-SM)342,345,500
  342 IF(DIST.GE.ROUT)GOTO 500
      NHIT=0
  345 NHIT=NHIT+1
      SM=ROUT
      LSURT(NHIT)=-LRO
      NASCT(NHIT)=NBO
      GOTO 500
```

are executed when point $\overline{XP}$ is at the RIN intersect of the present body and the next intersect does not occur at the same point. These statements test the ROUT intersect distance against the distance to the last body tested. If ROUT is greater, control is passed to the beginning of the DO loop to test the next body. If ROUT is smaller, it is compared with the distance that $\overline{XP}$ has travelled in the region. If the distance of $\overline{XP}$ is greater, control is passed to the beginning of the DO loop to test the next body. If the distance is smaller, the variable SM is equated to ROUT to represent the shortest distance to the next intersect, the body number of the intersect is recorded, and the variable LSURT is set negative for the intersect occurring at a leaving surface. Control is then passed to the beginning of the DO loop to test the next body.

The statements

```
C
C15   POINT XP AT ROUT OF BODY
C
  350 IF(ABS(RIN-SM).GT.SM*1.0E-6)GOTO 351
      RIN=SM
      IJK=LRIN+NBO-1
      ASTER(IJK)=RIN
      GOTO 355
```

are executed when the RIN intersect of the body under test does not occur on the same body at which point $\overline{XP}$ is now located. These statements determine if the present RIN intersect and the intersect of the last body tested occur at the same point. If they do, the intersect of the previous body is assigned to RIN.

The statements

```
  351 IF(RIN-SM)352,355,500
  352 IF(DIST.GE.RIN)GOTO 340
      NHIT=0
  355 NHIT=NHIT+1
      SM=RIN
      LSURT(NHIT)=LRI
      NASCT(NHIT)=NBO
  500 CONTINUE
```

are executed when the intersect of the last body and RIN of the present body do not occur simultaneously. These statements test the RIN intersect against the intersect distance to the last body tested. If RIN is greater, control is returned to the beginning of the DO loop to test the next body. If RIN is less, it is compared with $\overline{XP}$. If $\overline{XP}$ is greater than or equal to RIN, control is passed to test the ROUT distance. If RIN is greater than $\overline{XP}$, the variable SM is equated to RIN to represent the shortest distance to the next intersect of the bodies tested. The body number of the intersect is assigned to NASC, and the variable LSURT is set positive for the intersect occurring at an entering surface. Control is then passed to the beginning of the DO loop to test the next body.

The statements

```
C
      IF(SM.LT.PINF)GOTO 530
C
C16   ERROR=NO INTERSECT
C
      WRITE (6,902)IR
      WRITE (6,903)XB,WB,KLOOP,NBO,LRI,LRO,NMIT,LOOP
      GOTO 700
```

determine if an intersect after $\overline{XP}$ was found after testing all of the bodies in the region. If no intersect was found, an error message is printed and control is passed to the error diagnostic print section.

The statements

```
C
C17   COMPUTE NEW COORDINATES OF POINT XP AND REVISE DISTANCE TRAVELLED
C
  530 S1=S1+SM-DIST
      DIST=SM
      XP(1)=XB(1)+SM*WB(1)
      XP(2)=XB(2)+SM*WB(2)
      XP(3)=XB(3)+SM*WB(3)
```

update the cumulative distance that $\overline{XP}$ has travelled to the next intersect, after all of the bodies in the region have been considered. The point $\overline{XP}$ is also relocated to the next intersect.

The statement

```
C
      IF(NASC.EQ.-2)RETURN
```

determines if this subroutine has been called by Subroutine CALC to find the normal distance. If the computations are for the normal distance, control is returned to Subroutine CALC. If this subroutine was called to find the distance to the next region, the following DO loop is executed to determine the region where point $\overline{XP}$ is now located.

The statement

```
C
C18    DETERMINE REGION THAT POINT XP NOW IN
C
       DO 640 NN=1,NHIT
```

begins a DO loop which will scan the table of hits prepared by the previous DO loop to determine the region where point $\overline{XP}$ is now located.

The statements

```
       NASC =NASCT(NN)
       LSURF=LSURT(NN)
       LTRUE=0
```

are used to assign the body number of the hit being considered to the variable NASC. The surface number of the hit is assigned to the variable LSURF, and the variable LTRUE is initialized to zero. LTRUE is set to one by Subroutine WOWI if $\overline{XP}$ is located in the region passed to Subroutine WOWI by Subroutine G1.

The statements

```
C
C19    COMPUTE LOCATION OF INTERSECTED BODY DATA
C
       LOC=LBODY+3*(NASC-1)
       LOC=LOC+1
C
C20    RETRIEVE LOCATIONS OF REGION ENTER/LEAVE TABLE FOR BODY
C
       CALL UN2(LOC,LENT,LEAV)
       LOC=LOC+1
C
C21    RETRIEVE NUMBER OF REGIONS IN ENTRY LIST AND EXIT LIST
C
       CALL UN2(LOC,NENT,NEAV)
```

retrieve the location of the body data for the body being tested; then, using this location, retrieve the locations of the entrance and exit lists. The total number of regions that $\overline{XP}$ can be in when it is either entering or leaving the present body is then retrieved.

The statements

```
C
C22   COMPUTE THE BEGIN AND END OF LIST
C
      IF(LSURF.LE.0)GOTO 600
      J1=LENT
      J2=LENT+NENT-1
      GOTO 610
  600 J1=LEAV
      J2=LEAV+NEAV-1
```

determine if $\overline{XP}$ is at an entering or leaving surface of the body under test. The variable J1 is assigned to the location of the start of entry (or exit) list, and the variable J2 is assigned to the location of the end of the entry (or exit) list.

The statements

```
C
C23   ANY REGIONS IN LIST OR IS RAY LEAVING RPP
C
  610 IRPRIM=MASTER(J2)
      IF(J1.LE.J2)GOTO 620
      IF(NASC.GT.NRPP)GOTO 700
      IF(LSURF)630,700,700
```

are used to assign to the variable IRPRIM the last region number from the region table for later use in the error diagnostic section if there is an error in the region table. The beginning of the enter (or leave) list is compared to the end of the enter (or leave) list. If there are entries in the list, control is passed to determine which region the ray is entering. No entries in the list indicate that the ray is leaving the RPP. Since solid numbers are always larger than the number of RPP's, a comparison is made between the variable NASC and the variable NRPP to verify that the ray is leaving the RPP. If the variable NASC does not represent the current RPP, there is an error, and control is transferred to the error diagnostic section. If the variable NASC does represent the RPP solid number, a further test is made to verify that the ray is leaving the RPP by testing the variable LSURF for a negative condition. If not negative, there is an error. If negative, control is transferred to find the number of the abutting RPP.

The statements

```
C
C24   DETERMINE REGION POINT XP NOW ENTERING
C
  620 DO 625 J=J1,J2
      IRPRIM=MASTER(J)
      CALL WOWI(IRPRIM,LSURF,NASC,LTRUE)
      IF(LTRUE.GT.0)GOTO 650
  625 CONTINUE
```

test all of the regions in the enter (or leave) list against the current location of point $\overline{XP}$ by calling Subroutine WOWI for each region until Subroutine WOWI returns with the variable LTRUE set to one. This indicates that point $\overline{XP}$ is in the region just passed to Subroutine WOWI.

The statement

```
C
C25   RAY LEAVING RPP
C
      IF(NASC.GT.NRPP)GOTO 640
```

is executed if the previous DO loop could not locate a region for point $\overline{XP}$. This statement therefore determines if the current body is an RPP or other body. If other than an RPP, control is transferred to examine the next hit in the hits table.

The statement

```
      IF(LSURF)630,700,640
```

is executed if the previous test determined that the current body was an RPP. This statement therefore tests the variable LSURF to determine if the ray is leaving or entering the current RPP. If leaving (LSURF negative), control is transferred to find the number of the abutting RPP. If entering, control is transferred to examine the next hit in the hits table.

The statement

```
  630 CALL RPP2(LSURF,XP,IRP)
```

is used when the ray is leaving the current RPP to compute the number of the RPP that the ray is entering.

The statements

```
      IF(IRP.GT.0)GOTO 631
      IRPRIM=0
      RETURN
```

test the RPP number that is returned by Subroutine RPP2. If Subroutine RPP2 returns with a zero value, the ray is to be terminated or there is no abutting RPP. Region variable IRPRIM is set to zero, and control is then returned to the calling program.

The statements

```
C
C26    RETRIEVE LOCATION/NUMBER OF REGION ENTER LIST
C      COMPUTE BEGINNING AND END OF LIST
C
  631 LTRUE=0
      LOC=LBODY+3*(IRP-1)
      LOC=LOC+1
      CALL UN2(LOC,LENT,LEAV)
      LOC=LOC+1
      CALL UN2(LOC,NENT,NEAV)
      J1=LENT
      J2=LENT+NENT-1
```

are executed when Subroutine RPP2 returns with the number of an abutting RPP. These statements set the variable LTRUE to zero and retrieve the location of the body data of the new RPP, from which the locations of the region entry list and the number of possible regions for the entry list are computed.

The statement

```
      IF(J1.GT.J2)GOTO 700
```

determines if there are any regions listed in the entry list. No entries indicate an error in the entry list.

The statements

```
C
C27    DETERMINE REGION POINT XP NOW ENTERING IN NEW RPP
C
      DO 632 J=J1,J2
      IRPRIM=MASTER(J)
      CALL WOWI(IRPRIM,LSURF,IRP,LTRUE)
      IF(LTRUE.GT.0)GOTO 650
  632 CONTINUE
  640 CONTINUE
      GOTO 700
```

test the region in the enter list against the current location of point $\overline{XP}$ by calling Subroutine WOWI for each region in the list until Subroutine WOWI returns with the variable LTRUE set to one. This indicates that point $\overline{XP}$ is in the region just passed to Subroutine WOWI. If no region is found from the enter list, control is returned to the beginning of the main DO loop to analyze the next hit in the table of hits. If no region has been found after analyzing all of the hits in the table of hits, an error has occurred.

The statement

```
C
C28   REGION POINT XP ENTERING HAS BEEN DETERMINED
C
  650 IF(IR.EQ.IRPRIM)GOTO 660
```

begins the section for testing the region where $\overline{XP}$ is located when that region has been located. This statement is used to determine if the region where point $\overline{XP}$ is now located is the same as the region at the last location of point $\overline{XP}$.

The statements

```
      IF(S1.EQ.0.)GOTO 660
      IF(S1.LT.0.)GOTO 700
      IF(ABS(S1).LE.1.0E-6)GOTO 660
```

are executed if point $\overline{XP}$ has entered a new region. These statements therefore test the distance travelled through the previous region to determine if it is zero, less than zero, or very nearly zero. If the distance is zero or very nearly zero, control is transferred to update the region identifier and continue the ray. If S1 is less than zero, an error has occurred.

The statements

```
      IF(IVOLUM.EQ.IYES)RETURN
      IF(ITESTG.EQ.IYES)RETURN
```

determine if either Subroutine VOLUM or Subroutine TESTG called Subroutine G1 by comparing the variables IVOLUM or ITESTG with the variable IYES. If IVOLUM=1 (IYES=1), Subroutine G1 was called by Subroutine VOLUM, and control is returned to Subroutine VOLUM; if ITESTG=1, Subroutine G1 was called by Subroutine TESTG, and control is returned to Subroutine TESTG.

The statements

```
C
C29   RETRIEVE SPACE AND COMPONENT CODE OF REGION
C
      LOC=LIRFO+IR-1
      CALL UN2(LOC,ICODE,IDENT)
      LOC=LIRFO+IRPRIM-1
      CALL UN2(LOC,ICODE1,IDENT1)
```

retrieve the component codes (ICODE and ICODE1) and the space codes (IDENT and IDENT1) for the region where point $\overline{XP}$ was previously located and for the new region where point $\overline{XP}$ is presently located.

The statement

```
      IF(IDENT.EQ.1)GOTO 655
```

determines if the space of the previous region is exterior volume by testing space code variable IDENT for a value of one.

The statements

```
      IF(IDENT.EQ.IDENT1)GOTO 660
      RETURN
```

are executed if the previous region of point $\overline{XP}$ was interior volume. These statements determine if the space codes of the previous region of point $\overline{XP}$ and the present region of $\overline{XP}$ are identical. If the space codes are identical, control is transferred to update the region identifier and continue the ray. If the space codes are not identical, control is returned to the calling program.

The statement

```
  655 IF(ICODE.NE.ICODE1)RETURN
```

is executed if it was determined that the previous region of point $\overline{XP}$ was exterior volume. This statement therefore compares the component codes of the two regions. If the component codes are different, control is transferred to the calling program.

The statements

```
  660 IR=IRPRIM
      GOTO 20
```

update the region identifier to the region where point $\overline{XP}$ is now located if the two regions are found to be identical, or if distance (S1) through the previous region is zero or very nearly zero, or if the space codes of the two regions for interior volume are equal, or if the component codes of the two regions for exterior volume are identical. After updating the region identifier to the present region, control is transferred to the beginning of this subroutine to continue tracing the ray through the region.

The statements

```
C
C30   START OF ERROR DIAGNOSTIC SECTION
C
  700 IERR=IERR+1
      WRITE (6,904)IERR
```

are the beginning of the section of the subroutine for diagnostic error printing. These statements are executed when an error has occurred in the subroutine. Therefore, the variable IERR for the number of errors in Subroutine G1 is increased by one, and an error message is printed out.

The statements

```
C
C31   COMPUTE GRID CELL NUMBER IF G1 NOT CALLED BY VOLUM OR TESTG
C
      IF(IVOLUM.EQ.IYES.OR.ITESTG.EQ.IYES)GOTO 705
      IH=ABS(H/CELSIZ )+.5
      IF(H.LT.0.)IH=-IH
      IV=ABS(V/CELSIZ )+.5
      IF(V.LT.0.)IV=-IV
      WRITE (6,905)IH,IV
```

compute and print out the horizontal cell number and the vertical cell number if this subroutine was not called by either Subroutine VOLUM or Subroutine TESTG.

The statement

```
 705 WRITE (6,906)J1,J2,LSURF,NASC,IR,SM,S1,WB,XB
```

is used to print out various program parameters of Subroutine G1 for error checking.

The statements

```
C
C32   COMPUTE COORDINATES OF XP AT TIME OF ERROR
C
      XBD(1)=XB(1)-DIST
      XBD(2)=XB(2)-DIST
      XBD(3)=XB(3)-DIST
      WRITE (6,907)XBD,DIST
```

compute the coordinates of a point relative to $\overline{XB}$, the location of the start of the ray in the region, and the distance, DIST, that $\overline{XP}$ has travelled into the region at the time of the error.

The statements

```
      WRITE (6,908)
      NN=NBODY+NRPP
```

print column headings for the error diagnostic table that is to be prepared by the following DO loop. The limit of the DO loop is also computed as the total of the number of bodies plus the number of the rectangular parallelepiped in the region.

The statement

```
C
C33   PRINT OUT PERTINENT DATA FOR ALL BODIES IN REGION INTERSECTED
C     BY RAY FOR ERROR ANALYSIS
C
      DO 750 I=1,NN
```

begins a DO loop which scans all of the solids in the region to find those intersected by the present ray.

The statements

```
      LOC=LIO+I-1
      CALL UN3(LOC,I1,I2,I3)
```

retrieve the KLOOP value that was stored for the solid earlier in the subroutine when the intersected solid was located and its RIN and ROUT distance computed and stored.

The statement

```
      IF(KLOOP.NE.I3)GOTO 750
```

determines if the current solid was intersected by the ray. If not, control is returned to the beginning of the DO loop to test the next solid.

The statements

```
      IJK=LRIN+I-1
      RIN=ASTER(IJK)
      IJK=LROT+I-1
      ROUT=ASTER(IJK)
```

retrieve the RIN and ROUT values of the solid when one is found that was intersected by the ray.

The statements

```
      IF(RIN.NE.ROUT)GOTO 710
      WRITE (6,910)RIN,I
      GOTO 750
```

compare RIN with ROUT for the current intersected solid. If the values are equal, an error has occurred and an error message is printed out. Control is then returned to the beginning of the DO loop to consider the next solid.

The statement

```
C
  710 IF(ABS(RIN).NE.PINF)GOTO 720
```

determines if the ray had an entry point on the solid. If there was a RIN of the solid, the program branches to compare RIN and the distance to the previous intersect before the error occurred.

The statement

```
      IF(ABS(ROUT)-PINF)740,750,740
```

is executed if the previous IF statement determined that the ray did not enter the solid. This statement therefore tests ROUT for an exit intersect. If there is no exit intersect, there is an error in the program, and control is returned to test the next solid. If an exit occurred, the ray originated within the solid, and control is passed to a WRITE statement for output of information on the solid.

The statements

```
  720 IF(RIN-DIST)730,744,745
  730 IF(ROUT-DIST)741,742,743
```

are executed if the solid has an entry intersect. The first statement compares RIN and the distance, DIST, to the previous intersect before the error occurred. If RIN is greater, the ray will hit the present solid. If RIN is equal to DIST, the ray was entering the present solid. If RIN is less than DIST, control is transferred to compare ROUT with DIST. The comparison of ROUT and DIST reveals that the ray was in the solid when the error occurred if ROUT is greater, or that the ray has hit the solid if ROUT is less, or that the ray is leaving the solid if ROUT and DIST are equal. Whichever condition has occurred is printed out, and control is transferred to test the next solid.

The statements

```
C
  740 WRITE (6,911)RIN,ROUT,I1,I2,I
      GOTO 750
  741 WRITE (6,912)RIN,ROUT,I1,I2,I
      GOTO 750
  742 WRITE (6,913)RIN,ROUT,I1,I2,I
      GOTO 750
  743 WRITE (6,914)RIN,ROUT,I1,I2,I
      GOTO 750
  744 WRITE (6,915)RIN,ROUT,I1,I2,I
      GOTO 750
  745 WRITE (6,916)RIN,ROUT,I1,I2,I
C
  750 CONTINUE
```

print out the applicable results of the previous diagnostic tests. After the printout, control is returned to the beginning of the DO loop to test the next body.

The statements

```
      WRITE (6,917)IERR,IERR,IERR,IERR
      IRPRIM=-1
```

print out the number of errors that have occurred in Subroutine G1 and set the region variable IRPRIM to -1 to indicate to the calling program than an error has occurred.

The statements

```
C
  800 IF(IERR.GE.NG1ERR)WRITE (6,918)NG1ERR
      RETURN
      END
C
```

compare the number of errors that have occurred in Subroutine G1 with the limit of Subroutine G1 errors. If the number of errors is greater than the limit, an error message is printed. Control is then returned to the calling program.

Subroutine WOWI(JREG,LSURF,NEX,LTRUE)

This subroutine is used to determine if a given point lies within a region when given a point, $\overline{XB}$, and a region, JREG. That is, WOWI determines which of the possible regions actually has been entered by the ray. The subroutine determines if given point $\overline{XB}$ satisfies the description of region JREG by testing each body in JREG against the following rules:

1. A (+) operator is valid if ROUT > 0 and RIN $\leq$ DIST < ROUT.

2. A (-) operator is valid if ROUT $\leq$ 0 or DIST < RIN or DIST $\geq$ ROUT.

3. An (OR) operator is valid only if every (+) and (-) within the (OR) statement is valid.

4. A region description containing one or more (OR) statements is satisfied if any one of the (OR) statements is valid.

5. A region description containing no (OR) statement is satisfied only if every (+) and (-) operator is valid.

6. A sufficient condition for point $\overline{XB}$ to be a region JREG is that the region description of JREG be satisfied. (Two regions cannot be satisfied for the same point.)

The statements

```
 DIMENSION MASTER(10000)
 COMMON ASTER(10000)
 COMMON/PAREM/XB(3),WB(3),IR
 COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
 COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
 COMMON/DAVIS/IGRID,LOOP,INORM
 COMMON/WHICH/NBO
```

dimension the MASTER array and pass information into and out of the subroutine.

The statement

```
 EQUIVALENCE(ASTER,MASTER)
```

sets the MASTER array equivalent to the ASTER array.

The statement

```
C
  901 FORMAT(1H0*32HERROR IN G1 AT 140     BAD ITYPE*5X,4HITY=*I5)
```

is a format statement for an error message to be used later in the subroutine.

The statement

```
C
      LOC=LREGD*JREG-1
```

is used to set LOC to the location of the region data in the MASTER array of the region to be considered.

The statements

```
C
C1     RETRIEVE NUMBER OF BODIES IN REGION AND LOCATION OF
C      OPERATOR/BODY LIST
C
       CALL UN2(LOC*LOCD,NC)
C
C2     RETRIEVE FIRST OPERATOR/BODY FROM LIST
C
       CALL UN2(LOCD*IOP,NBO)
```

retrieve the location of the region data, the number of descriptors in the region being considered, the operator of the first descriptor, and the solid number of the first descriptor.

The statements

```
      N=1
      IOPER=IOP
```

initialize to the first descriptor and assign the operator of the first descriptor to a different variable name which will represent the operator part of each different descriptor.

The statements

```
C
C3      RETRIEVE ENTER AND EXIT SURFACE NUMBERS AND NUMBER OF LAST RAY
C
   10 ITEMP=LIO+NBO-1
      CALL UN3(ITEMP,LRI,LRO,LOOP)
```

retrieve the entering surface number, the leaving surface number, and the value at location LOOP.

The statements

```
C
C4      RETRIEVE BODY TYPE AND LOCATION OF DATA
C
        ITEMP=LBODY+3*(NBO-1)
        CALL UN2(ITEMP,ITYPE,LOCDA)
```

compute and retrieve, from the body data, the type of body (ITYPE) that the solid number (NBO) now being processed represents.

The statement

```
        IF(LOOP.NE.KLOOP)GOTO 30
```

is used to determine if RIN and ROUT for the current solid have already been computed. If not, control is transferred to compute RIN and ROUT for the body type under consideration.

The statement

```
C
        IF(ITYPE.GT.11)GOTO 40
```

is used to determine if the number of the body type is valid. If it is not, control is transferred to print an error.

The statements

```
C
C5      RETRIEVE RIN AND ROUT FOR CURRENT BODY
C
        IJK=LRIN+NBO-1
        RIN=ASTER(IJK)
        IJK=LROT+NBO-1
        ROUT=ASTER(IJK)
```

retrieve the values of RIN and ROUT if they have already been computed.

The statement

```
      IF(ITYPE.LT.10)GOTO 310
```

determines if the type of body for the retrieved RIN and ROUT is a torus or an arbitrary surface, since these two bodies can have more than one RIN/ROUT set.

The statements

```
C
CA    IS NEXT RIN/ROUT SET REQUIRED FOR TOR OR ARS
C
      IF(ROUT.LT.0.)GOTO 400
      IF(DIST.LE.ROUT)GOTO 310
```

determine if the ray misses the torus or arbitrary surface. If the ray intersects, a test is made to determine if the computed distance is greater than the present ROUT for this body. A ROUT less than the distance indicates that the point along the ray is past the recorded RIN/ROUT set. The subroutine therefore computes the next RIN/ROUT set, if any. If the distance is less than the stored ROUT, control is transferred to compute the distance to the next intersection.

The statements

```
C
   30 LRI=1
      LRO=1
      ITY=ITYPE+1
      IF(ITY.GE.1.AND.ITY.LE.12)GOTO 100
```

initialize the entering and leaving surface numbers to one, increment the body type to avoid a zero index in the following computed GOTO statement, and test the body number to determine if it is a valid number. If not, an error is printed out.

The statements

```
 40 IERR=IERR+1
    WRITE (6,901) ITYPE
    RETURN
```

are used to print out an error message if the body number is not valid, and then to the calling program Subroutine G1.

The statements

```
C
C7     COMPUTE RIN/ROUT FOR CURRENT BODY
C
C             RPP BOX SPH RCC REC TRC ELL RAN ARB TEC TOR ARS
  100 GOTO(110,120,130,140,150,160,170,180,190,200,210,220),ITY
  110 CALL RPP(NBO)
      GOTO 300
  120 CALL BOX
      GOTO 300
  130 CALL SPH
      GOTO 300
  140 CALL RCC
      GOTO 300
  150 CALL REC
      GOTO 300
  160 CALL TRC
      GOTO 300
  170 CALL ELL
      GOTO 300
  180 CALL RAW
      GOTO 300
  190 CALL ARB
      GOTO 300
  200 CALL TEC
      GOTO 300
  210 CALL TOR
      GOTO 300
  220 CALL ARS
```

are used to transfer control to a statement that calls the body subroutine for the body being considered.

The statements

```
C
  300 IJK=LIO+NBO-1
      MASTER(IJK)=KLOOP+I15*(LRO+64*LRI)
```

are used to pack and store into one word the entering surface number, the leaving surface number, and the current value of KLOOP at a location according to the body number.

The statements

```
C
C8     DETERMINE CORRECT RIN/ROUT AND STORE IN ASTER ARRAY
C
  310 IF(ROUT.LE.0.)GOTO 330
      IF(ABS(RIN-DIST).GT.DIST*1.0E-6)GOTO 320
      RIN=DIST
      GOTO 330
C
  320 IF(ABS(ROUT-DIST).LE.DIST*1.0E-6)ROUT=DIST
C
  330 IJK=LRIN+NBO-1
      ASTER(IJK)=RIN
      IJK=LROT+NBO-1
      ASTER(IJK)=ROUT
```

determine if the ray hits the body. If not, control is transferred to store RIN and ROUT for a miss. If the ray intersects the body, it is determined whether the next intersect was a RIN or ROUT. RIN or ROUT are equated to the distance, and the values of RIN and ROUT are stored in the ASTER array according to the body number.

The statement

```
C
C9     TEST CONDITIONS FOR POINT XB IN REGION UNDER TEST
C
  400 IF(IOPER.GT.4)GOTO 500
```

determines if the operator of the present body is a (-) by testing for a greater than 4 condition. The numbers 5-8 represent (-) operators; 1-4 represent (+) operators. The numbers 1 or 5 also represent (OR) operators.

The statements

```
C
C10    (+) OPERATOR TEST   RIN.LE.DIST.LT.ROUT   POINT XB IN BODY
C
       IF(RIN.GT.DIST)GOTO 700
       IF(DIST-ROUT)600,700,700
```

are executed if it was determined by the previous test that the operator was a (+). These statements test for a valid (+) operator by testing for RIN $\leq$ distance > ROUT and ROUT > 0. If these conditions are not satisfied, the program branches to test for an (OR) operator condition. If these conditions are satisfied, the program branches to check the next body in the description.

The statements

```
C
C11    (-) OPERATOR TEST   ROUT.LE.0 OR DIST.LT.RIN OR DIST.GE.ROUT
C      POINT XB OUTSIDE OF BODY
C
  500  IF(ROUT.LE.0.)GOTO 600
       IF(DIST.LT.RIN)GOTO 600
       IF(DIST.EQ.RIN)GOTO 700
       IF(DIST.LT.ROUT)GOTO 700
```

are executed for a (-) operator condition to test for ROUT $\leq$ 0 or distance < RIN or distance $\geq$ ROUT. If any one of these conditions is true, the program branches to check the next body in the description. If one of the conditions is not true and RIN $\leq$ distance < ROUT, the program branches to test for an (OR) operator condition.

The statements

```
C
C12    CHECK NEXT BODY IN OPERATOR/BODY LIST
C
  600  IF(N.GE.NC)GOTO 800
       N=N+1
       LOCD=LOCD+1
       CALL UN2(LOCD,IOPER,NBO)
       IF(IOPER.EQ.1.OR.IOPER.EQ.5)GOTO 800
       GOTO 10
```

are executed if either a valid (+) or (-) operator was found. These statements check if the last body in the region is being considered and, if true, branch to indicate that point $\overline{XB}$ lies in the region presently being considered. If the last body in the region is not being considered, the next body and operator are located and retrieved and tested for a 1 or 5 to determine if there is an (OR) operator. If so, point $\overline{XB}$ lies in the present region since a region description is satisfied if any one of the (OR) statements is valid. If there is no (OR) operator, control is transferred to the beginning of the subroutine to process the next descriptor.

The statements

```
C
C13    (OR) OPERATOR TEST
C      ALL (+) OR (-) IN (OR) SERIES MUST BE VALID
C
  700 IF(IOP.NE.1.AND.IOP.NE.5) RETURN
      IF(N.GE.NC)RETURN
      N=N+1
      DO 710 NN=N,NC
      LOCD=LOCD+1
      CALL UN2(LOCD,IOPER,N40)
      IF(IOPER.EQ.1.OR.IOPER.EQ.5)GOTO 720
  710 CONTINUE
      RETURN
  720 N=NN
      GOTO 10
```

are executed if the test on a (+) or (-) operator was not valid. These statements test for an (OR) operator; if not an (OR) operator, control is returned to the calling program with the result that point $\overline{XB}$ is not in the present region. However, if the test reveals an (OR) operator, and the present body is not the last descriptor in the region, the next body and operator are located and retrieved. The new operator is tested for an (OR) operator; if there is an (OR) operator, the search of the descriptors in the region is continued until either a (+) or (-) operator is found, or until all of the descriptors have been tested. If a (+) or (-) operator is not found, control is returned to the calling program with the result that point $\overline{XB}$ is not in the present region. If a (+) or (-) operator is located for a descriptor within the present region, control is transferred to the beginning of the subroutine to process this next descriptor.

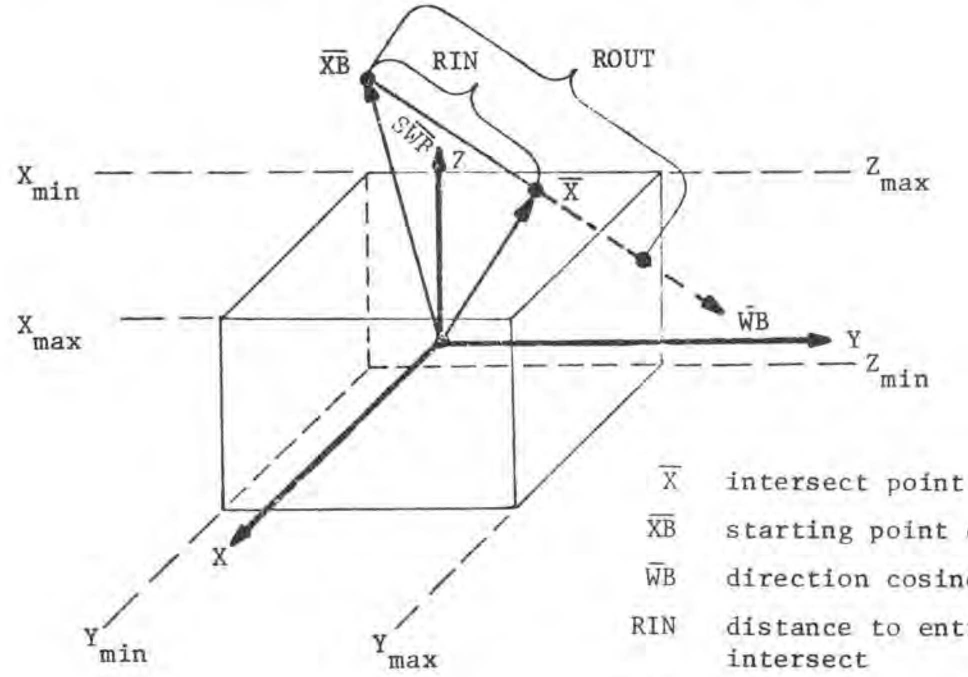
The statements

```
C
C14    POINT XB WITHIN CURRENT REGION. LTRUE = 1
C
  800 LTRUE=LTRUE+1
      RETURN
      END
C
C
```

set LTRUE to one and return to Subroutine G1 with the result that point $\overline{XB}$ lies within the present region being considered.

Subroutine RPP(NBO)

This subroutine is used to compute the intersection of a ray with a rectangular parallelepiped as depicted in the following figure:



$\overline{X}$	intersect point on body
$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 60. Rectangular Parallelepiped

The statements

```
 DIMENSION PR(6),LR(6),XS(6),LST(6)
 DIMENSION MASTER(10000)
 COMMON ASTER(10000)
 COMMON/PAREM/XB(3),WB(3),IR
 COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
 COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and pass information into and out of this subroutine.

The statement

```
 EQUIVALENCE (MASTER,ASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statement

```
C
  901 FORMAT(1H0,12HERROR IN RPP/4H L =,I10,5X,4HNBO=,I10,5X,3HIR=,
     1  I10/4H XB=,3E20.10/4H WB=,3E20.10/4H PR=,6E20.10/4H LR=,6I10)
```

is a format statement for an error message later in the subroutine.

The statements

```
C
C1     SET UP SIX MEMBER ARRAY TO REPRESENT COORDINATE PAIRS
C
       LST(1)=1
       LST(2)=1
       LST(3)=2
       LST(4)=2
       LST(5)=3
       LST(6)=3
```

are used to set up a six member array to denote the x, y, z pairs, respectively, of the RPP.

The statements

```
       L=0
       PR(1)=0.
       PR(2)=0.
```

are used to initialize subroutine variables to zero for later use in the subroutine.

The statements

```
C
C2    RETRIEVE THE SIX BOUNDARIES OF THE RPP
C
      DO 10 I=1,6
      XS(I)=S(NBO,I)
   10 CONTINUE
```

constitute a loop which retrieves, using Function S(I,N), the location in the ASTER array of the coordinates of the six sides of the rectangular parallelepiped and stores them in a six-element array as follows:

$XS(1) = X_{min}$

$XS(2) = X_{max}$

$XS(3) = Y_{min}$

$XS(4) = Y_{max}$

$XS(5) = Z_{min}$

$XS(6) = Z_{max}$

The statement

```
C
      DO 100 I=1,6
```

is used to start a loop which will compute the ray intersections (if any) and the distances from the origin of the ray to the intersections with the RPP.

The statement

```
      II=LST(I)
```

is used to assign the value of LST(I) or face pair number to the variable II for use as a subscript.

The statement

```
      TEMP=XS(I)-XB(II)
```

is used to compute the value of $(\overline{XS}_I - \overline{XB}_J)$ for the equation

$$\boxed{S_I = \frac{\overline{XS}_I - \overline{XB}_J}{\overline{WB}_J}} \qquad \begin{array}{cc} \underline{J} & \underline{I} \\ 1 & 1,2 \\ 2 & 3,4 \\ 3 & 5,6 \end{array} \tag{2}$$

where

S_I is the distance from $\overline{XB}$ to the intersection point

$\overline{XS}_I$ is a bounding plane of the RPP

$\overline{XB}_J$ is the $\overline{XB}_x$, $\overline{XB}_y$ or $\overline{XB}_z$ coordinate of $\overline{XB}$

$\overline{WB}_J$ is the $\overline{WB}_x$, $\overline{WB}_y$ or $\overline{WB}_z$ direction cosine of $\overline{WB}$

The statements

```
      IF(WB(II)) 20,100,30
   20 IF(TEMP)40,100,100
   30 IF(TEMP.LE.0.)GOTO 100
   40 TRY=TEMP/WB(II)
```

are used to determine if the direction of the ray is away from or toward the plane in question. If away from the plane (S_I is negative), a return is made to the beginning of the loop, and a new plane is then considered. If toward the plane (S_I is positive), the distance to the plane is computed by Equation (2).

The statements

```
      DO 60 J=1,3
      IF(J.EQ.II)GOTO 60
C
C3    COMPUTE INTERSECT/PLANE COORDINATE
C
      XRY=XB(J)+TRY*WB(J)
C
C4    DETERMINE IF INTERSECT OCCURS WITHIN BOUNDARY OF PLANE
C
      IF((XS(2*J-1)-XRY)*(XRY-XS(2*J)).LT.0.)GOTO 100
   60 CONTINUE
```

constitute a loop which computes the intersect point $\overline{XI}_C$ for the plane being considered from the equation

$$\boxed{\overline{XI}_C = \overline{XB}_C + \overline{WB}_C \cdot S_I} \tag{3}$$

This intersect point is then tested to determine if it lies within the confines of the bounding planes according to the equations (for an intersect point on an X plane)

$$\boxed{\begin{array}{l} Y_{min} \leq Y_C \leq Y_{max} \\ Z_{min} \leq Z_C \leq Z_{max} \end{array}} \tag{4}$$

where Y_C and Z_C are the y-z coordinates of the intersect point on the X plane being considered.

The statements

```
      L=L+1
C
C*    COMPUTE DISTANCE TO INTERSECT POINT
C
      PR(L)=TRY
      LR(L)=I
      IF(L.EQ.2)GOTO 130
      IF(L.LT.2)GOTO 100
```

are used to store the distance, S_I, to the intersect point on the plane and the surface number of the plane with the intersect point. A test is also made to determine if two planes of the RPP have been intersected (the maximum possible number). If not, a return is made to the beginning of the loop and a new plane is considered. If two planes have been intersected , an exit from the loop is performed to compute the values of RIN and ROUT.

The statements

```
      WRITE (6,901)L,NBO,IR,XB,WB,PR,LR
      ROUT=-PINF
      RETURN
```

are used to output an error message along with some program values.

The statement

```
  100 CONTINUE
```

is used to return control to the beginning of the loop if all six bounding planes have not been considered. If the six planes have been considered, control is passed out of the loop to the next statement.

The statement

```
      GOTO 160
```

is used to pass control to another part of the subroutine after all six bounding planes of the RPP have been considered.

The statement

```
C
  130 IF(ABS(PR(1)-PR(2)).LE.PR(1)*1.0E-6)GOTO 200
```

is used to determine if the distance between the intersect points is less than a very small minimum value, such as an intersect taking place on a corner. If the distance is large enough, RIN and ROUT and the entering and leaving surface numbers are determined.

The statement

```
IF(PR(1)-PR(2))140,180,150
```

is used to determine which of the intersect points is RIN and ROUT.

The statements

```
C
C6    COMPUTE RIN, ROUT, AND SURFACE NUMBERS OF INTERSECTS
C
  140 RIN=PR(1)
      LRI=LR(1)
      ROUT=PR(2)
      LRO=LR(2)
      RETURN
  150 RIN=PR(2)
      LRI=LR(2)
      ROUT=PR(1)
      LRO=LR(1)
      RETURN
```

are two identical groups of statements which compute the values of RIN, ROUT, the entering surface number, and the leaving surface number, depending upon the results of the previous test that determined which of the intersect points was RIN and which was ROUT. When the computation is complete for either condition, control is returned to the calling program.

The statements

```
C
  160 IF(L.GE.1)GOTO 180
C
C7    ASSIGN VALUE TO ROUT FOR NO INTERSECTION
C
  170 ROUT=-PINF
      RETURN
```

are used to determine if no intersection took place or if one intersection took place. If no intersection took place, ROUT is set to an extremely large negative value and control is returned to the calling program. If one intersection occurred, control is passed to the next group of statements for computation of RIN and ROUT.

The statements

```
C
C8     RAY ORIGINATES WITHIN RPP
C
  180 RIN=-PINF
      LRI=0
      ROUT=PR(1)
      LRO=LR(1)
      RETURN
```

are used to compute the values of RIN and ROUT when only one intersection occurred, which means that the ray originates within the RPP. Therefore, RIN is set to an extremely large negative value. The entering surface number is set to zero, the value of ROUT is recorded, and the leaving surface number is recorded. Control is then returned to the calling program.

The statements

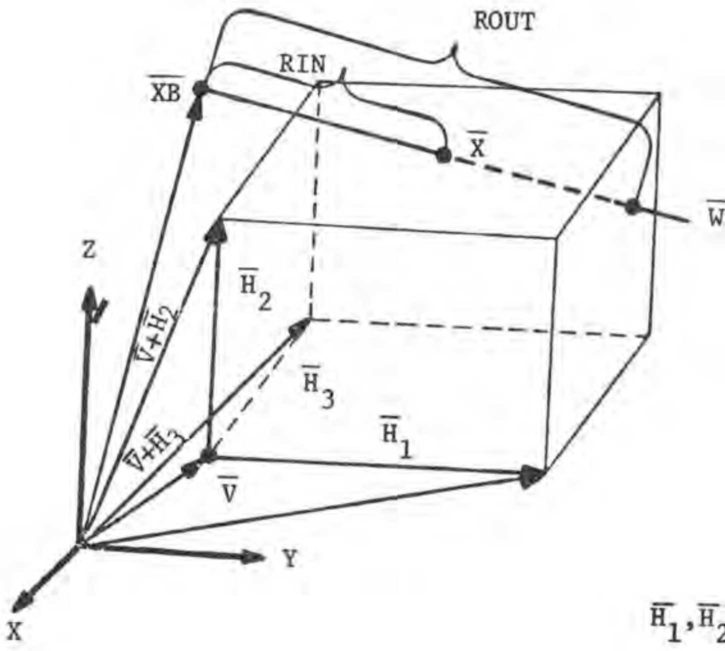
```
C
C9     DETERMINE IF RAY ORIGINATES WITHIN RPP OR MISSES
C
  200 DO 220 J=1,3
      IF(XB(J).LT.XS(2*J-1))GOTO 170
      IF(XB(J).GT.XS(2*J))GOTO 170
  220 CONTINUE
      GOTO 180
      END
C
C
```

comprise a loop to determine if the origin of the ray is within the RPP. If not, control is passed out of the loop where a miss is recorded. If the ray originates within the RPP, control is passed to record the intersect point and surface numbers affected.

Subroutine BOX

Subroutine BOX calculates the intersection distances (if any) of a ray in space and a box. The intersection distances, RIN and ROUT, are calculated by solving simultaneously the ray equation and the equations defining the sides of the box.

The box is defined by a vertex, $\overline{V}$, and three mutually perpendicular length vectors. These three vectors represent the height, width, and length of the box as shown in Figure 61.



$\overline{X}$	intersect point of ray
$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	vertex of box
$\overline{H}_1, \overline{H}_2, H_3$	three mutually perpendicular length vectors of the box
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 61. Box

The statements

```
      DIMENSION MASTER(10000)
      COMMON ASTER(10000)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1 LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and pass information into and out of this routine. The statement

```
      EQUIVALENCE (MASTER,ASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statements

```
C
C1    RETRIEVE LOCATION OF BOX VERTEX AND H1 COORDINATES
C
      CALL UN2(LOCDA,IV,IH1)
      LOC=LOCDA+1
C
C2    RETRIEVE LOCATION OF BOX H2 AND H3 COORDINATES
C
      CALL UN2(LOC,IH2,IH3)
```

are used to retrieve the pointers to the locations in the ASTER array of the box vertex and triplet coordinate values.

The statements

```
      RIN=-PINF
      ROUT=PINF
```

are used to initialize the variable RIN to an extremely large negative value and the variable ROUT to an extremely large positive value.

The statement

```
DO 105 I=1,3
```

is used to start a loop which will compute the ray intersection with the three pairs of opposite faces of the box.

The statements

```
   IF(I-2)11,12,13
11 II=2
   GOTO 14
12 II=1
   GOTO 14
13 II=3
```

are used to assign a value to the variable II according to the pass number of the loop. The variable II is used later in the loop to compute the face number of the box.

The statements

```
14 A=0.
   VP=0.
   W=0.
```

are used to initialize the variables A, VP, and W to zero.

The statements

```
C
C3    COMPUTE VECTOR DOT PRODUCTS
C
      DO 15 J=1,3
      JV=IV+J
      JA=IH1+J
      VP=VP+(ASTER(JV-1)-XB(J))*ASTER(JA-1)
      W=W+WB(J)*ASTER(JA-1)
      A=A+ASTER(JA-1)**2
   15 CONTINUE
```

are used to compute the vector dot products required for Equations (8) and (9).

$$\boxed{S_V^i = (\overline{V}-\overline{XB}) \cdot \overline{H}_i \Big/ (\overline{WB}\cdot\overline{H}_i)} \qquad (8)$$

$$\boxed{S_{V+H}^i = S_V^i + \overline{H}_i \cdot \overline{H}_i \Big/ (\overline{WB}\cdot\overline{H}_i)} \qquad (9)$$

Thus

$$VP = (\overline{V}-\overline{XB}) \cdot \overline{H}_i = (\overline{V}_x-\overline{XB}_x)\cdot\overline{H}_{ix} + (\overline{V}_y-\overline{XB}_y)\cdot\overline{H}_{iy} + (\overline{V}_z-\overline{XB}_z)\cdot\overline{H}_{iz}$$

$$W = \overline{WB} \cdot \overline{H}_i = \overline{WB}_x\cdot\overline{H}_{ix} + \overline{WB}_y\cdot\overline{H}_{iy} + \overline{WB}_z\cdot\overline{H}_{iz}$$

$$A = \overline{H}_i \cdot \overline{H}_i = \overline{H}_{ix}^2 + \overline{H}_{iy}^2 + \overline{H}_{iz}^2$$

The statement

```
IF(W)30,20,40
```

is used to test the value of $\overline{WB}\cdot\overline{H}_i$ and branch accordingly. If $\overline{WB}\cdot\overline{H}_i$ is zero, the ray is parallel to the faces in question, and an intersection will not occur.

The statements

```
20 IF(-VP.LT.0.)GOTO 200
   IF(-VP-A)100'100'200
```

are used to test whether an intersection is possible with any other face. If $(\overline{V}-\overline{XB})\cdot\overline{H}_i$ is positive, an intersection with the box is not possible, and this routine is exited via Statement 200.

The statements

```
C
C4    COMPUTE ROUT
C
   30 CP=VP/W
      LO=2*II-1
```

are used to compute ROUT_i for the face in question by Equation(8)and LO if $\overline{WB}\cdot\overline{H_i}$ is negative. LO will have the following values:

LO = 3 for H1
LO = 1 for H2
LO = 5 for H3

The statement

```
      IF(CP.LE.0.)GOTO 200
```

is used to test the sign of ROUT_i. If ROUT is negative, the box is not located on the proper side of the ray initiation point, and the routine is exited via Statement 200.

The statements

```
C
C5    COMPUTE RIN
C
      CM=(VP+A)/W
      LI=LO+1
      GOTO 60
```

are used to compute RIN_i for the faces in question by Equation(9)and LI. LI will have the following values:

LI = 4 for H1
LI = 2 for H2
LI = 6 for H3

Execution transfers to Statement 60, and the computed values of ROUT and RIN are compared with the previously computed values.

The statements

```
C
CA     COMPUTE ROUT
C
   40 CP=(VP+A)/W
      LO=2*II
      IF(CP.LE.0.)GOTO 200
C
C7     COMPUTE RIN
C
      CM=VP/W
      LI=LO-1
```

are used to compute $ROUT_i$ by Equation(9) and RIN_i by Equation(8) if $\overline{WB}\cdot\overline{H}_i$ is positive. If $ROUT_i$ is negative the routine is executed via Statement 200. LO and LI are given the following values:

LO = 4, LI = 3 for H1
LO = 2, LI = 1 for H2
LO = 6, LI = 5 for H3

The statements

```
  60 IF(ROUT.LE.CP)GOTO 80
     ROUT=CP
     LRO=LO
  80 IF(RIN.GE.CM)GOTO 100
     RIN=CM
     LRI=LI
```

are used to compare the computed values of RIN and ROUT with the previously assigned values. If the computed value of ROUT is greater than the previously assigned value, the new value is retained and LRO is updated to identify the face associated with ROUT. Similarly, the computed value of RIN is retained if it is less than the previously assigned value, and LRI is updated.

The statements

```
 100 IH1=IH2
     IH2=IH3
 105 CONTINUE
```

are used to increment the face description vectors to select two new faces, and the computations are repeated until all faces have been examined.

The statements

```
      IF(ABS(RIN-ROUT).LE.ROUT*1.0E-6)GOTO 200
      IF(RIN.LT.ROUT)RETURN
```

are used to compare the values of ROUT and RIN. If RIN has the same value as ROUT, the routine is exited via Statement 200. If RIN is less than ROUT, the execution returns to the calling routine with RIN and ROUT set at the computed values.

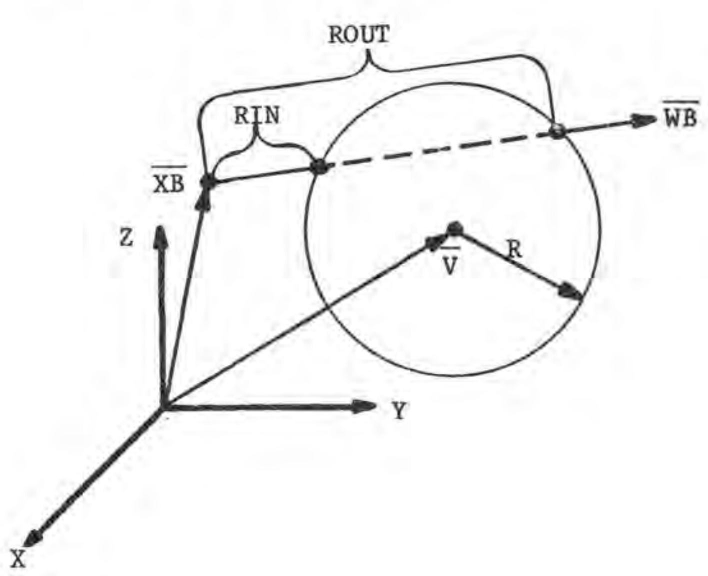
The statements

```
C
  200 RIN=PINF
      ROUT=-PINF
      RETURN
```

are used to set RIN at an extremely large positive value and ROUT at an extremely large negative value and to return execution to the calling routine.

Subroutine SPH

The subroutine is used to compute the intersection of a ray with a sphere as depicted in the following figure:



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	center of sphere
R	radius of sphere
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 62. Sphere

The statements

```
 COMMON ASTER(10000)
 COMMON/PAREM/XB(3),WB(3),IR
 COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
 COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to pass information into and out of this subroutine.

The statement

```
C
C1    RETRIEVE LOCATION OF SPH VERTEX AND RADIUS
C
      CALL UN2(LOCDA,ITEMP,I2)
```

is used to retrieve the locations at which the sphere's vertex and radius values are stored in the ASTER array.

The statement

```
      R=ASTER(I2)
```

equates the retrieved value for the sphere's radius to R.

The statements

```
ITEMP=ITEMP+1
DX=XB(1)-ASTER(ITEMP-1)
DY=XB(2)-ASTER(ITEMP)
DZ=XB(3)-ASTER(ITEMP+1)
```

are used to compute the coordinates of the vector $\overline{DX}$ where

$$\overline{DX}_x = \overline{XB}_x - \overline{V}_x$$

$$\overline{DX}_y = \overline{XB}_y - \overline{V}_y$$

$$\overline{DX}_z = \overline{XB}_z - \overline{V}_z$$

The statement

```
B=DX*WB(1)+DY*WB(2)+DZ*WB(3)
```

is then used to compute the dot product $\overline{DX}\cdot\overline{WB}$, which is used in Equations (14) and (15)

$$S = -B \pm \sqrt{B^2-C}$$

where

$$\boxed{RIN = -B - \sqrt{B^2-C}} \qquad (14)$$

$$\boxed{ROUT = -B + \sqrt{B^2-C}} \qquad (15)$$

where $B = \overline{DX}\cdot\overline{WB}$

The statement

```
C=DX*DX+DY*DY+DZ*DZ-R*R
```

is used to compute the value of C according to Equation $C = \overline{DX}^2-R^2$.

The statement

```
      DIS=B*B-C
```

is used to compute the value B^2-C for later use within the subroutine when the square root function will be used.

The statements

```
      IF(C.GT.0.)GOTO 10
C
C2    RAY ORIGINATES WITHIN SPHERE
C
      RIN=-PINF
      ROUT=SQRT(DIS)-B
      RETURN
```

are used to test if the origin of the ray is inside or outside of the sphere. If C is negative the origin is within the sphere. RIN is therefore set to an extremely large negative value, ROUT is computed and control is returned to the calling program. If the origin of the ray is external to the sphere, control is transferred to Statement 10 in the subroutine.

The statements

```
C
   10 IF(DIS.GT.0.)GOTO 20
C
C3    RAY MISSES SPHERE
C
      RIN=PINF
      ROUT=-PINF
      RETURN
```

are used to test if the ray intersects the sphere. If not, RIN is set to an extremely large positive value, ROUT is set to an extremely large negative value, and control is returned to the calling program. If the ray intersects the sphere, control is transferred to Statement 20 in the subroutine.

The statements

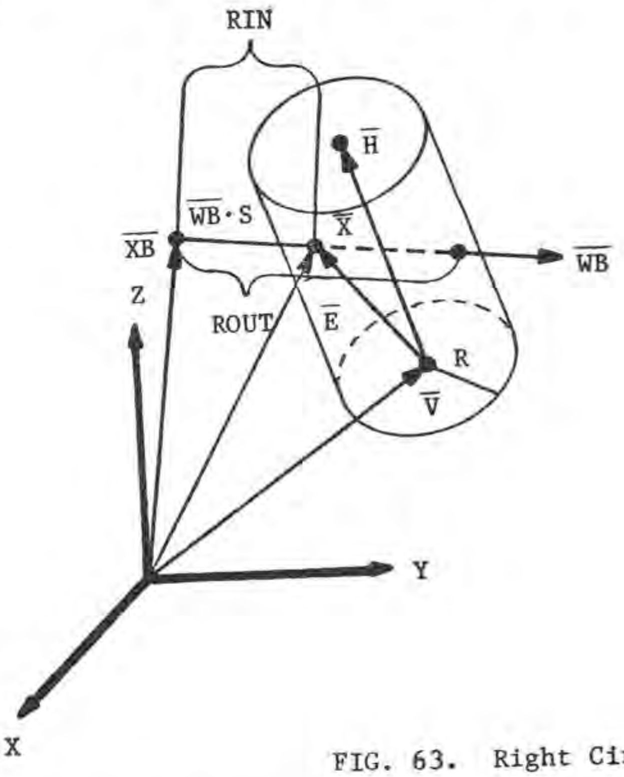
```
C
C4    RAY INTERSECTS SPHERE
C
```

```
20 DIS=SQRT(DIS)
   RIN=-B-DIS
   ROUT=-B+DIS
   RETURN
```

are used to compute the values of RIN and ROUT for a ray originating outside the sphere and intersecting the sphere. When RIN and ROUT are computed, control is returned to the calling program.

Subroutine RCC

This subroutine is used to compute the intersections of a ray with a right circular cylinder as depicted in the following figure:



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	vertex of RCC
$\overline{H}$	height vector of RCC
R	radius of RCC

FIG. 63. Right Circular Cylinder

The statements

```
DIMENSION V(3),H(3)
DIMENSION MASTER(10000)
COMMON ASTER(10000)
COMMON/PAREM/XB(3),WB(3),IR
COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1 LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and pass information into and out of this subroutine.

The statement

```
EQUIVALENCE (ASTER,MASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statement

```
C
C1     RETRIEVE LOCATION OF RCC VERTEX AND HEIGHT VECTOR COORDINATES
C
       CALL UN2(LOCDA,IV,IH)
```

is used to retrieve from the ASTER array the pointer to the location of the coordinates for the vertex and the height vector of the right circular cylinder.

The statement

```
C
C2     RETRIEVE LOCATION OF RADIUS
C
       IRR=MASTER(LOCDA+1)
```

is used to retrieve from the MASTER array the location of the radius of the right circular cylinder.

The statements

```
C
C3     RETRIEVE COORDINATES OF VERTEX AND HEIGHT VECTOR
C
       H(1)=ASTER(IH)
       H(2)=ASTER(IH+1)
       H(3)=ASTER(IH+2)
       V(1)=ASTER(IV)
       V(2)=ASTER(IV+1)
       V(3)=ASTER(IV+2)
```

are used to retrieve and fill two three-element arrays with the value of the RCC height vector and vertex in terms of the x, y, and z components.

The statement

```
C
C4     RETRIEVE RADIUS
C
       R=ASTER(IRR)
```

is used to retrieve the value of the radius of the right circular cylinder.

The statements

```
       RIN=-PINF
       ROUT=PINF
```

are used to initialize RIN to an extremely large negative value and ROUT to an extremely large positive value.

The statement

```
C
C5      COMPUTE R SQUARED
C
        RSQ=R*R
```

is used to compute the value of R^2.

The statements

```
        LRO=0
        LRI=0
        TOP=0.
        POT=0.
```

are used to initialize the variables LRO, LRI, TOP, and POT to zero.

The statements

```
C
C6      COMPUTE VECTOR DOT PRODUCTS
C
        HH=H(1)*H(1)+H(2)*H(2)+H(3)*H(3)
        VPH=H(1)*(V(1)-XB(1))+H(2)*(V(2)-XB(2))+H(3)*(V(3)-XB(3))
        WH=WB(1)*H(1)+WB(2)*H(2)+WB(3)*H(3)
C
C7      COMPUTE COEFFICIENT OF S SQUARED
C
        DEN=HH-WH*WH
```

are used to compute the dot products $\overline{H}\cdot\overline{H}$, $\overline{H}\cdot(\overline{V}-\overline{XB})$, and $\overline{WB}\cdot\overline{H}$ respectively, and to compute the value of the numerator of τ for use in solving Equations (22), (27), and (29). Equation (22) is the equation for the distances to the intersect points on an infinite right circular cylinder. Equations (27) and (29) are the equations for the distances to the intersect points with the planar surfaces.

$$\boxed{\begin{aligned} & S^2\left[1-\frac{(\overline{WB}\cdot\overline{H})^2}{\overline{H}\cdot\overline{H}}\right]-2S\left[\frac{\overline{WB}\cdot\overline{H}\cdot(\overline{XB}-\overline{V})\cdot\overline{H}}{\overline{H}\cdot\overline{H}}-\overline{WB}\cdot(\overline{WB}-\overline{V})\right] \\ & +\left[(\overline{XB}-\overline{V})(\overline{XB}-\overline{V})-\frac{[(\overline{XB}-\overline{V})\cdot\overline{H}]^2}{\overline{H}\cdot\overline{H}}-R^2\right]=0 \end{aligned}} \tag{22}$$

or

$$\boxed{\tau S^2 - 2\lambda' S + \mu' = 0} \tag{23}$$

If

$$\lambda = \lambda'/\tau \ , \ \mu = \mu'/\tau$$

Then

$$S^2 - 2\lambda S + \mu = 0$$

Solving for S gives

$$S = \lambda \pm \sqrt{\lambda^2 - \mu}$$

Thus, possible intersect points (RIN and ROUT) for an infinite right circular cylinder are:

$$\boxed{RIN = \lambda - \sqrt{\lambda^2 - \mu}} \tag{24}$$

$$\boxed{ROUT = \lambda + \sqrt{\lambda^2 - \mu}} \tag{25}$$

The possible intersect points (RIN and ROUT) for two planar surfaces are:

$$\boxed{S_V = \frac{\overline{H}\cdot(\overline{V}-\overline{XB})}{\overline{WB}\cdot\overline{H}}} \tag{27}$$

for the $\overline{V}$ plane and

$$\boxed{S_{V+H} = \frac{\overline{H}\cdot(\overline{V}-\overline{XB}) + \overline{H}\cdot\overline{H}}{\overline{WB}\cdot\overline{H}}} \tag{29}$$

for the $\overline{V}+\overline{H}$ plane

RIN = S_V and ROUT = S_{V+H} for $\overline{WB}\cdot\overline{H} > 0$

RIN = S_{V+H} and ROUT = S_V for $\overline{WB}\cdot\overline{H} < 0$

For $\overline{WB}\cdot\overline{H} = 0$, the ray is parallel to the two planes, and no intersections with the planes will occur.

The statements

```
      DO 10 I=1,3
      TOP=TOP+WB(I)*(XB(I)-V(I))
      POT=POT+(XB(I)-V(I))**2
   10 CONTINUE
```

constitute a loop which computes the values of $\overline{WB}\cdot(\overline{XB}-\overline{V})$ and $(\overline{XB}-\overline{V})^2$, which will be used later in the subroutine to solve Equation (22) for S.

The statements

```
      AMBD=-HH*TOP-WH*VPH
      UM=(POT-RSQ)*HH-VPH**2
```

are used to compute the value of the numerator of λ', and the value of μ' for solving Equation (22).

The statement

```
      IF(WH)40,70,50
```

is used to test the value of the dot product $\overline{WB}\cdot\overline{H}$ to determine if the ray is parallel to the two planes ($\overline{WB}\cdot\overline{H} = 0$).

The statements

```
C
C8    SOLVE FOR RIN AND ROUT OF PLANE INTERSECTIONS
C
   40 CP=VPH/WH
      CM=(VPH+HH)/WH
      LCP=1
      LCM=2
      GOTO 60
   50 CP=(VPH+HH)/WH
      CM=VPH/WH
      LCM=1
      LCP=2
```

are made up of two groups of instructions which calculate the distance from the ray origin to the intersect points on the two planes. The first group calculates the distances in terms of a negative $\overline{WB}\cdot\overline{H}$, and the second group calculates the distance in terms of a positive $\overline{WB}\cdot\overline{H}$. The surface numbers of these two planes are also stored in a temporary location for later use in the subroutine.

The statement

```
   60 IF(CP)300,80,80
```

is used to determine if the ray is moving away from the two planes of the cylinder. If the distance to the intersect point of the ROUT plane is negative, the ray misses the RCC and control is passed to another area of the subroutine where the total miss condition is recorded.

The statements

```
70 CP=PINF
   CM=-CP
   IF(VPH.GT.0.)GOTO 300
   IF(HH+VPH)300,90,90
```

are used if the ray is parallel to the two planes of the RCC. Temporary locations of RIN and ROUT for the two plane surfaces are first set to a miss condition. Next, a test is made to determine if the $\overline{V}$ plane is above the ray. If it is, an exit is made to the end of the subroutine where a total miss condition is recorded before return to the calling program. If the $\overline{V}$ plane is not above the ray, another test is performed to determine if the ray is above the $\overline{V}+\overline{H}$ plane. If it is, a total miss condition is recorded as before. If not, control is transferred to determine if the ray intersects the quadratic surface.

The statement

```
80 IF(ABS(DEN).GE.1.0E-6)GOTO 90
```

is used to determine if the absolute value of τ of Equation (22) is less than or equal to a very small value, which indicates that the ray is parallel to the sides.

The statements

```
   R1=-PINF
   R2=PINF
   GOTO 100
```

are used to set temporary subroutine variables for RIN and ROUT to an extremely large negative number and an extremely large positive number, respectively, if the previous test determined that the ray was parallel to the $\overline{H}$ vector. Control is then transferred around the calculations for intersections with the quadratic surface.

The statements

```
90 R1=0.
   R2=0.
```

initialize two temporary subroutine variables to zero.

The statements

```
AMBDA=AMBD/DEN
UMU=UM/DEN
DISC=AMBDA**2-UMU
```

are used to compute the value of λ ($\lambda = \lambda'/\tau$), μ ($\mu = \mu'/\tau$) and the quantity $\lambda^2-\mu$.

The statement

```
IF(DISC.LE.0.)GOTO 300
```

is used to determine if $\lambda^2-\mu \leq 0$, which would mean that there are no intersections with the quadratic surface.

The statement

```
SD=SQRT(DISC)
```

is used to obtain the square root of $(\lambda^2-\mu) > 0$.

The statements

```
C
C*    SOLVE FOR RIN AND ROUT OF QUADRATIC INTERSECTIONS
C
      R1=AMBDA-SD
      R2=AMBDA+SD
```

are used to compute the distances from the ray origin to the intersect points on an infinite cylinder by Equations (24) and (25).

The statement

```
100 IF(CM.GT.R1)GOTO 110
```

is used to determine if the entrance of the ray into the RCC is on one of the planes or on the quadratic surface.

The statements

```
    RIN=R1
    LRI=3
    GOTO 120
```

are used to assign RIN a value and to assign the surface number of the entry point for the entrance of the ray on the quadratic surface.

The statements

```
110 RIN=CM
    LRI=LCM
```

are used to assign RIN a value, and to assign the surface number of the entry point for the entrance of the ray into one of the planar surfaces.

The statement

```
120 IF(CP.LE.R2)GOTO 130
```

is used to determine if the exit of the ray from the RCC is on one of the planes or on the quadratic surface.

The statements

```
    ROUT=R2
    LRO=3
    GOTO 200
```

are used to assign ROUT a value, and to assign the surface number of the exit point for the exit of the ray out of the quadratic surface. A branch is then made around the exit of a ray from a planar surface.

The statements

```
  130 ROUT=CP
      LRO=LCP
```

are used to assign ROUT a value and to assign the surface number of the exit point for the exit of the ray out of the quadratic surface.

The statement

```
  200 IF(ABS(ROUT-RIN).LE.ROUT*1.0E-5)GOTO 300
```

is used to determine if the distance between the intersect points is less than or equal to a very small minimum value, such as an intersect taking place at a corner. If the distance is extremely small,control is passed to the end of the subroutine where a miss is recorded.

The statement

```
      GOTO(210,210,220),LHO
```

is used to transfer program control depending upon the surface number of the exit of the ray. If the ray exits from one of the two planar surfaces, control is transferred to determine if the intersection with the planar surface lies within the circular cross section of the cylinder. If the ray exits from the quadratic surface, control is transferred to determine if the intersection with the quadratic surface lies between the two planar surfaces.

The statements

```
C
C10   DETERMINE IF ROUT INTERSECTS PLANE WITHIN CYLINDER CROSS-SECTION
C
  210 F1=DEN*ROUT**2-2.*AMBD*ROUT+UM
      IF(F1)250,250,300
```

are used to determine if ROUT from a planar surface lies within the circular cross section of the cylinder by evaluating Equation (23) for ROUT=S. The result must be less than or equal to zero for a valid intersection.

The statements

```
C
C11   DOES ROUT INTERSECT OF QUADRATIC SURFACE OCCUR BETWEEN PLANES
C
  220 F1=ROUT*WH-VPH
      IF(F1)300,250,230
```

are used to evaluate and test Equation (27) for ROUT=S if ROUT is from the quadratic surface. If the value of the expression is negative, the intersection lies below the bottom planar surface. If the value of the expression is zero, ROUT occurs at a corner and control is transferred to determine the locations of RIN. If the expression is positive, control is transferred to determine if ROUT is below the top planar surface.

The statement

```
  230 IF(F1.GT.HH) GOTO 300
```

is used to determine if ROUT is above the top planar surface. If it is, control is transferred to record a miss.

The statement

```
  250 GOTO(260,260,270),LRI
```

is used to transfer program control depending upon the number of the surface which the ray enters. If the ray enters one of the two planar surfaces control is transferred to determine if the intersection with the planar surface lies within the circular cross section of the cylinder. If the ray enters the quadratic surface, control is transferred to determine if the intersection with the quadratic surface lies between the two planar surfaces.

The statements

```
C
C12   DETERMINE IF RIN INTERSECTS PLANE WITHIN CYLINDER CROSS-SECTION
C
  260 F1=DEN*RIN**2-2.*AMBD*RIN+UM
      IF(F1)310,310,300
```

are used to determine if RIN on a planar surface lies within the circular cross section of the cylinder by evaluating Equation (23) for RIN=S. The result must be less than or equal to zero for a valid intersection.

The statements

```
C
C13   DOES RIN INTERSECT OF QUADRATIC SURFACE OCCUR BETWEEN PLANES
C
  270 F1=RIN*WH-VPH
      IF(F1)300,310,280
```

are used to evaluate and test Equation (27) for RIN=S if RIN is on the quadratic surface. If the expression is negative, the intersection lies below the bottom planar surface. If the expression is zero, RIN occurs at a corner, and control is transferred to the calling program. If the expression is positive, control is transferred to determine if RIN is below the top planar surface.

The statement

```
  280 IF(F1.LE.HH)GOTO 310
```

is used to determine if RIN is above the top planar surface. If so, control is returned to the calling program.

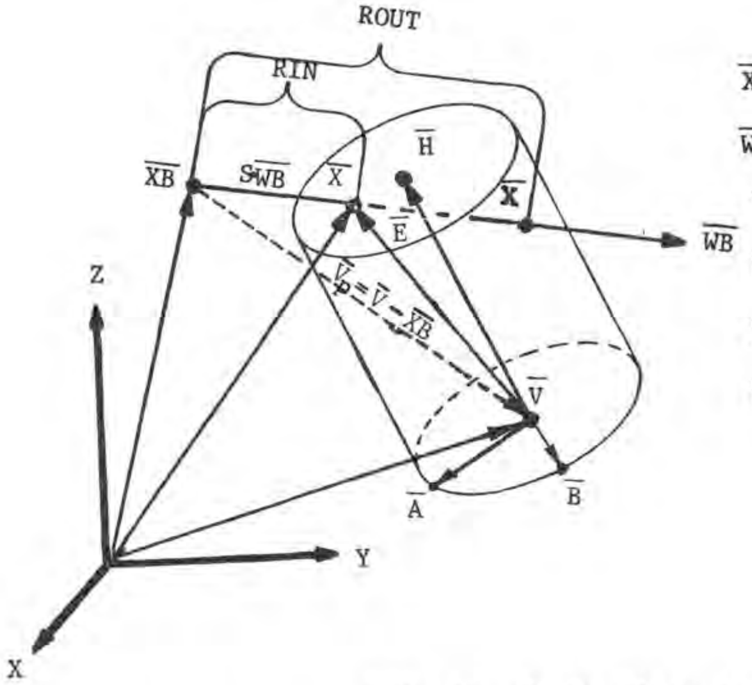
The statements

```
C
C14   RAY MISSES BODY
C
  300 RIN=PINF
      ROUT=-PINF
      LRO=0
      LRI=0
  310 RETURN
      END
C
C
```

are used to record a miss of the RCC. RIN is set to an extremely large positive value, and ROUT is set to an extremely large negative value. Also, the surface entry and exit codes are set to zero. Return is then made to the calling program.

Subroutine REC

This subroutine is used to compute the intersection of a ray with a right elliptic cylinder as depicted in the following figure:



$\overline{XB}$	a fixed point on the ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	vertex of REC
$\overline{H}$	height vector of ray
$\overline{A}$	semi-major axis
$\overline{B}$	semi-minor axis

FIG. 64. Right Elliptic Cylinder

The statements

```
      DIMENSION V(3),H(3),A(3),B(3)
      COMMON ASTER(10000)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMCN/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTHIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and pass information into and out of this subroutine.

The statements

```
C
C1    RETRIEVE LOCATION OF REC VEXTEX AND HEIGTH VECTOR COORDINATES
C
      CALL UN2(LOCDA,IV,IH)
      LOC=LOCDA+1
C
C2    RETRIEVE LOCATION OF REC COORDINATES FOR AXES
C
      CALL UN2(LOC,IA,IB)
```

are used to retrieve from the MASTER array the locations at which the right elliptic cylinder's vertex, height vector, semi-major axis, and semi-minor axis coordinates are stored.

The statements

```
C
C3      RETRIEVE COORDINATES OF VERTEX, HEIGHT VECTOR, SEMI-MAJOR AXIS
C       AND SEMI-MINOR AXIS
C
        V(1)=ASTER(IV)
        V(2)=ASTER(IV+1)
        V(3)=ASTER(IV+2)
        H(1)=ASTER(IH)
        H(2)=ASTER(IH+1)
        H(3)=ASTER(IH+2)
        A(1)=ASTER(IA)
        A(2)=ASTER(IA+1)
        A(3)=ASTER(IA+2)
        B(1)=ASTER(IB)
        B(2)=ASTER(IB+1)
        B(3)=ASTER(IB+2)
```

are used to retrieve and fill four three-element arrays with the x, y, and z coordinates of the REC vertex, height vector, semi-major axis, and semi-minor axis.

The statements

```
        RIN=-PINF
        ROUT=PINF
        LRO=0
        LRI=0
```

initialize RIN and ROUT to a large negative and large positive value, respectively, and initialize the entering surface number and exit surface number variables to zero.

The statements

```
C
C4    COMPUTE DOT PRODUCTS OF A.A AND B.B
C
      AA=A(1)*A(1)+A(2)*A(2)+A(3)*A(3)
      BB=B(1)*B(1)+B(2)*B(2)+B(3)*B(3)
```

are used to compute the dot products $\overline{A}\cdot\overline{A}$ for the semi-major axis and $\overline{B}\cdot\overline{B}$ for the semi-minor axis.

The statements

```
C
C5    COMPUTE (V-XB) FOR X,Y,Z COORDINATES
C
      V1XB1=V(1)-XB(1)
      V2XB2=V(2)-XB(2)
      V3XB3=V(3)-XB(3)
```

are used to compute $\overline{V}_x - \overline{XB}_x$, $\overline{V}_y - \overline{XB}_y$, and $\overline{V}_z - \overline{XB}_z$, which defines the origin of the ray, $\overline{WB}$, with respect to the vertex of the REC.

The statements

```
C
C6    TRANSFORM XB(X,Y) TO THE COORDINATES OF THE REC
C
      VPA=V1XB1*A(1)+V2XB2*A(2)+V3XB3*A(3)
      VPB=V1XB1*B(1)+V2XB2*B(2)+V3XB3*B(3)
```

are used to compute the components of $\overline{V}_P$ with respect to $\overline{A}$ and $\overline{B}$ for subsequent solutions for S.

The statements

```
C
C7    TRANSFORM WB(X,Y) TO THE COORDINATES OF THE REC
C
      WBA=WB(1)*A(1)+WB(2)*A(2)+WB(3)*A(3)
      WBB=WB(1)*B(1)+WB(2)*B(2)+WB(3)*B(3)
```

are used to compute the components of $\overline{WB}$ with respect to $\overline{A}$ and $\overline{B}$ for subsequent solutions for S.

The statements

```
WBAWBA=WBA*WBA
WBBWBB=WBB*WBB
```

are used to compute the quantities $(\overline{WB}\cdot\overline{A})^2$ and $(\overline{WB}\cdot\overline{B})^2$.

The statements

```
AAAA=AA*AA
BBBB=BB*BB
```

are used to compute the quantities $(\overline{A}\cdot\overline{A})^2$ and $(\overline{B}\cdot\overline{B})^2$.

The statement

```
AMBD=WBA*VPA*BBBB+WBB*VPB*AAAA
```

is used to compute the coefficient, λ', of 2S from Equation (35). Equation (31) is the general form of an elliptic cylinder along the $\overline{H}$-axis.

$$\boxed{\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1} \tag{31}$$

or

$$b^2x^2 + a^2y^2 = a^2b^2 \tag{32}$$

where x^2 and y^2 are in the REC coordinate system. Therefore,

and $\quad a^2 = \overline{A}\cdot\overline{A}$ and $b^2 = \overline{B}\cdot\overline{B}$

The origin of the ray $\overline{XB}$ and the direction cosines of the ray $\overline{WB}$ can be transformed into the coordinate system of the REC, where $\overline{V}$ now becomes the origin, the x-axis is along the semi-major axis, the y-axis is along the semi-minor axis, and the z-axis is coincident with the $\overline{H}$ vector, by the dot products

$$\overline{VP}_A = (\overline{V} - \overline{XB}) \cdot \overline{A}$$
$$\overline{VP}_B = (\overline{V} - \overline{XB}) \cdot \overline{B}$$
$$\overline{W}_A = \overline{WB} \cdot \overline{A}$$
$$\overline{W}_B = \overline{WB} \cdot \overline{B}$$

The intersection of the transformed ray from the dot products is

$$\boxed{\overline{E} = \overline{VP} + S\cdot\overline{W}} \tag{33}$$

which on substituting into Equation (32) yields

$$b^2(\overline{VP}_A + \overline{W}_A \cdot S)^2 + a^2(\overline{VP}_B + \overline{W}_B \cdot S)^2 = a^2b^2 \tag{34}$$

Expanding Equation (34) where $a^2 = \overline{A} \cdot \overline{A}$ and $b^2 = \overline{B} \cdot \overline{B}$ gives

$$\boxed{\begin{aligned} &S^2\left[\left(\overline{B}\cdot\overline{B}\right)\left(\overline{W}_A\right)^2 + \left(\overline{A}\cdot\overline{A}\right)\left(\overline{W}_B\right)^2\right] \\ &-2S\left[\left(\overline{B}\cdot\overline{B}\right)\left(\overline{VP}_A\cdot\overline{W}_A\right) + \left(\overline{A}\cdot\overline{A}\right)\left(\overline{VP}_B\cdot\overline{WB}\right)\right] \\ &+\left[\left(\overline{B}\cdot\overline{B}\right)\left(\overline{VP}_A\right)^2 + \left(\overline{A}\cdot\overline{A}\right)\left(\overline{VP}_B\right)^2 - \left(\overline{A}\cdot\overline{A}\right)\left(\overline{B}\cdot\overline{B}\right)\right] = 0 \end{aligned}} \tag{35}$$

which is of the form

$$\boxed{\tau S^2 - 2\lambda' S + \mu' = 0}$$

where

$$\lambda = \lambda'/\tau \text{ and } \mu = \mu'/\tau$$

Therefore,

$$S^2 - 2\lambda S + \mu = 0$$

Solving for S

$$S = \lambda \pm \sqrt{\lambda^2 - \mu}$$

Thus, possible intersect points (RIN and ROUT) for an infinite right elliptic cylinder are:

$$\boxed{\text{RIN} = \lambda - \sqrt{\lambda^2 - \mu}} \qquad (37)$$

$$\boxed{\text{ROUT} = \lambda + \sqrt{\lambda^2 - \mu}} \qquad (38)$$

For $\lambda^2 - \mu \leq 0$, no intersection with the cylinder is possible. Also, $(b^2 \cdot \overline{W}_A{}^2 + a^2 \cdot \overline{W}_B{}^2) \leq 10^{-6}$ implies that the ray is parallel to the $\overline{H}$ vector, and intersections will occur only with the planar surfaces.

The possible intersect points (RIN and ROUT) for two planar surfaces are:

$$\boxed{S_V = \frac{\overline{VP}_H}{\overline{W}_H}} \qquad (39)$$

for the $\overline{V}$ plane, and

$$\boxed{S_{V+H} = S_V + \frac{(\overline{H} \cdot \overline{H})}{\overline{W}_H}} \qquad (40)$$

for the $\overline{V}+\overline{H}$ plane.

RIN = S_V and ROUT = S_{V+H} for $\overline{WB} \cdot \overline{H} > 0$.

RIN = S_{V+H} and ROUT = S_V for $\overline{WB} \cdot \overline{H} < 0$.

For $\overline{WB}\cdot\overline{H} = 0$ the ray is parallel to the two planes, and no intersections with the planes will occur.

If RIN or ROUT is an intersection with the quadratic surface, the intersection point must lie between the two planar surfaces.

Therefore,

$$\boxed{0 \le \left(S\cdot\overline{W}_H - \overline{VP}_H\right) \le \left(\overline{H}\cdot\overline{H}\right)} \tag{41}$$

where S is RIN or ROUT.

If RIN or ROUT is an intersection with a planar surface the intersection point must lie within the elliptic cross section.

Therefore,

$$\boxed{S^2 - 2\lambda S + \mu \le 0} \tag{42}$$

where S is RIN or ROUT.

The statement

```
UM=BBBB*VPA*VPA+AAAA*VPB*VPB-AAAA*BBBB
```

is used to compute the value of μ' for solving Equation (36).

The statement

```
DEN=WBA*WBA*BBBB+WBB*WBB*AAAA
```

is used to compute the value of τ which is the coefficient of S^2 from Equation (35).

The statement

```
IF(ABS(DEN).LE.1.0E-6)GOTO 10
```

is used to determine if the absolute value of τ of Equation (36) is less than or equal to a very small value, which indicates that the ray is parallel to the axis.

The statements

```
      AMBDA=AMBD/DEN
      UMU=UM/DEN
      DISC=AMBDA**2-UMU
```

are used to compute the value of λ where $\lambda = \lambda'/\tau$ and μ where $\mu = \mu'/\tau$, and to compute the quantity $\lambda^2-\mu$.

The statement

```
      IF(DISC.LE.0.)GOTO 300
```

is used to determine if the quantity $\lambda^2-\mu \leq 0$, which would mean that no intersection with the quadratic surface is possible.

The statements

```
C
CA    COMPUTE THE INTERSECT POINTS ON THE QUADRATIC SURFACE
C
      SD=SQRT(DISC)
      R1=AMBDA-SD
      R2=AMBDA+SD
      GOTO 20
```

are used to compute the square root of the $(\lambda^2-\mu) \geq 0$ and the distances from the ray origin to the intersect points on an infinite right elliptic cylinder by Equations (37) and (38).

The statements

```
   10 R1=-PINF
      R2=PINF
```

are used to set temporary subroutine variables for RIN and ROUT to an extremely large negative number and an extremely large positive number, respectively, if it was previously determined that the ray was parallel to the $\overline{H}$ vector.

The statements

```
   20 HH=H(1)*H(1)+H(2)*H(2)+H(3)*H(3)
      WH=WB(1)*H(1)+WB(2)*H(2)+WB(3)*H(3)
      VPH=V1XB1*H(1)+V2XB2*H(2)+V3XB3*H(3)
```

are used to compute the dot products $\overline{H}\cdot\overline{H}$, $\overline{WB}\cdot\overline{H}$, and $(\overline{V}-\overline{XB})\cdot\overline{H}$ for use in solving for the intersect distances to the planar surface.

The statement

```
C
C9    DETERMINE IF RAY PARALLEL TO PLANAR SURFACES
C
      IF(WH)40,70,50
```

is used to test the results of the dot product $\overline{WB}\cdot\overline{H}$ to determine the direction of the ray with respect to the $\overline{H}$ vector ($\overline{WB}\cdot\overline{H} < 0$ for a downward direction; $\overline{WB}\cdot\overline{H} > 0$ for an upward direction; $\overline{WB}\cdot\overline{H} = 0$ for perpendicular).

The statement

```
   40 IF(VPH.GE.0.)GOTO 300
```

is used to determine if the ray (moving in the opposite direction with respect to $\overline{H}$) will intersect the surface containing $\overline{V}$. A positive result of the test would result in a negative distance. Therefore, control is passed to another area of the subroutine where the total miss condition is recorded. A negative result of the test would result in a positive distance, which means the ray intersects the surface plane containing $\overline{V}$.

The statements

```
C
C10   COMPUTE THE INTERSECT POINTS ON THE PLANAR SURFACES
C
      CP=VPH/WH
      CM=(VPH+HH)/WH
      LCP=1
      LCM=2
      GOTO 100
```

are used to store the surface intersect numbers of RIN and ROUT and to compute the distance from the ray origin to the intersect points on the two planes for a ray moving in an opposite direction with respect to the vector $\overline{H}$. The program then branches to determine if the RIN and ROUT intersect points first occur at a planar surface or at the quadratic surface.

The statements

```
   50 VPHHH=VPH+HH
      IF(VPHHH.LE.0.)GOTO 300
```

are used to compute the quantity $\overline{H}\cdot(\overline{V}-\overline{XB}) + \overline{H}\cdot\overline{H}$ and then to determine if the ray will intersect the upper planar surface of the REC. A negative or zero value of the IF statement would result in a negative distance. Therefore, control is passed to the end of the subroutine where the total miss condition is recorded. A positive value in the test would result in a positive distance which means the ray intersects the upper planar surface.

The statements

```
      CP=VPHHH/WH
      CM=VPH/WH
      LCM=1
      LCP=2
      GOTO 100
```

are used to compute the distance to the planar intersect points for a ray moving in an upward direction with respect to the vector $\overline{H}$. The surface numbers of the intersected planes are also stored for later use before branching to determine if the RIN or ROUT intersect points occur at a planar surface or at the quadratic surface.

The statements

```
 70 CP=PINF
    CM=-CP
```

are used to set the ray entry distance to an extremely large negative value and the exit distance to an extremely large positive value, if the ray is parallel to the planar surfaces of the REC.

The statement

```
  100 IF(CM.GT.R1)GOTO 110
```

is used to determine if the entry point of the ray first occurs at a planar surface or on the quadratic surface.

The statements

```
C
C11   RIN FOR THE QUADRATIC SURFACE
C
      RIN=R1
      LRI=3
      GOTO 120
```

are used to assign RIN a value and to assign the surface number of the entry point when contact first occurs with the quadratic surface.

The statements

```
C
C12   RIN FOR A PLANAR SURFACE
C
  110 RIN=CM
      LRI=LCM
```

are used to assign RIN a value and to assign the surface number of the entry point when contact first occurs with a planar surface.

The statement

```
 120 IF(CP.LE.R2)GOTO 130
```

is used to determine if the exit point of the ray first occurs at a planar surface or on the quadratic surface.

The statements

```
C
C13   ROUT FOR THE QUADRATIC SURFACE
C
      ROUT=R2
      LRO=3
      GOTO 200
```

are used to assign ROUT a value and to assign the surface number of the exit point for the exit of the ray from the quadratic surface. A branch is then made around the exit of a ray from a planar surface.

The statements

```
C
C14   ROUT FOR A PLANAR SURFACE
C
  130 ROUT=CP
      LRO=LCP
```

are used to assign ROUT a value and to assign the surface number of the exit point for the exit of the ray from the quadratic surface.

The statement

```
  200 IF(ABS(ROUT-RIN).LE.ROUT*1.0E-5)GOTO 300
```

is used to determine if the distance between the intersect points is less than or equal to a very small minimum value, such as an intersect taking place at a corner. If the distance is extremely small, control is passed to the end of the subroutine where a miss is recorded.

The statement

```
      GOTO(210,210,220),LRO
```

is used to transfer program control depending upon the surface number where the ray exit occurs. If the ray exits from one of the two planar surfaces, control is transferred to determine if the intersection with the planar surface lies within the elliptic cross section of the cylinder. If the ray exits from the quadratic surface, control is transferred to determine if the intersection with the quadratic surface lies between the two planar surfaces.

The statements

```
C
C15   DETERMINE IF ROUT OF PLANAR SURFACE OCCURS WITHIN ELLIPTIC
C     CROSS-SECTION
C
  210 F1=DEN*ROUT**2-2.*AMBO*ROUT+UM
      IF(F1)250,250,300
```

are used to determine if ROUT from a planar surface lies within the elliptic cross section of the cylinder by evaluating Equation (42) for ROUT=S. The result must be less than or equal to zero for a valid intersection.

The statements

```
C
C16   DETERMINE IF ROUT OF QUADRATIC OCCURS BETWEEN PLANAR SURFACES
C
  220 F1=ROUT*WH-VPH
      IF(F1)300,250,230
```

are used to evaluate and test Equation (41) for ROUT=S if ROUT is from the quadratic surface. If the expression F1 is negative, the intersection lies below the bottom planar surface. If F1 is zero, ROUT occurs at a corner, and control is transferred to determine the location of RIN. If F1 is positive, ROUT is above the bottom planar surface, and control is transferred to determine if ROUT is below the top planar surface.

The statement

```
  230 IF(F1.GT.HH)GOTO 300
```

is used to determine if ROUT is above the top planar surface. If so, control is transferred to record a miss.

The statement

```
  250 GOTO(260,260,270),LRI
```

is used to transfer program control depending upon the surface number of the entrance of the ray. If the ray enters one of the two planar surfaces, control is transferred to determine if the intersection with the planar surface lies within the circular cross-section of the cylinders. If the ray enters the quadratic surface, control is transferred to determine if the intersection with the quadratic surface lies between the two planar surfaces.

The statements

```
C
C17   DETERMINE IF RIN OF PLANE WITHIN ELLIPTIC CROSS SECTION
C
  260 F1=DEN*RIN**2-2.*AMBD*RIN+UM
      IF(F1)310,310,300
```

are used to determine if RIN on a planar surface lies within the elliptic cross section of the cylinder by evaluating Equation (42) for RIN=S. The result must be less than or equal to zero for a valid intersection.

The statements

```
C
C18   DETERMINE IF RIN OF QUADRATIC SURFACE BETWEEN PLANAR SURFACES
C
  270 F1=RIN*WH-VPH
      IF(F1)300,310,280
```

are used to evaluate and test Equation (41) for RIN=S if RIN is on the quadratic surface. If the expression is negative, the intersection lies below the bottom planar surface. If the expression is zero, RIN occurs at a corner, and control is transferred to the calling program. If the expression is positive, control is transferred to determine if RIN is below the top planar surface.

The statement

```
  280 IF(F1.LE.HH)GOTO 310
```

is used to determine if RIN is above the top planar surface. If it is, control is returned to the calling program.

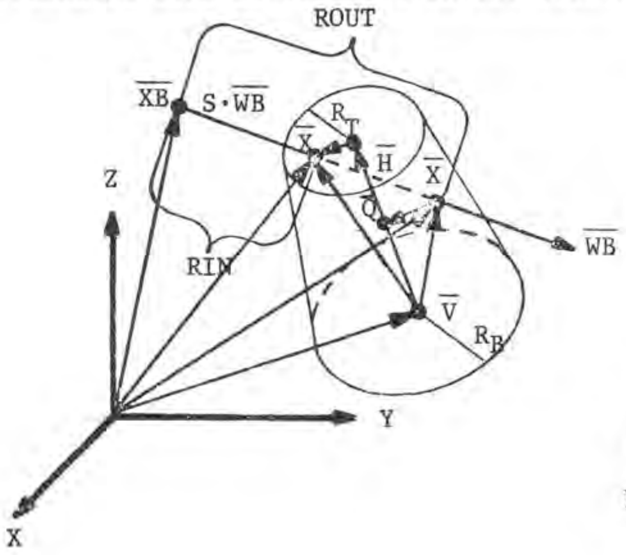
The statements

```
C
C19   RAY MISSES BODY
C
  300 RIN=PINF
      ROUT=-PINF
      LRI=0
      LRO=0
  310 RETURN
      END
C
C
```

are used to record a miss of the REC. RIN is set to an extremely large positive value, ROUT is set to an extremely large negative value, and the surface entry and exit numbers are set to zero. Return is then made to the calling program.

Subroutine TRC

This subroutine is used to compute the intersection of a ray with a right truncated cone as depicted in the following figure:



$\overline{V}$	vertex of the TRC
$\overline{H}$	height vector of the TRC
R_T	radius at $\overline{V}+\overline{H}$
R_B	radius at $\overline{V}$
$\overline{XB}$	a fixed point on the ray
$\overline{WB}$	direction cosines of the ray
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 65. Right Truncated Cone

The statements

```
      DIMENSION V(3),H(3)
      DIMENSION MASTER(10000)
      COMMON ASTER(10000)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and to pass information into and out of this subroutine.

The statement

```
      EQUIVALENCE(MASTER,ASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statements

```
C
C1     RETRIEVE LOCATION OF TRC VERTEX AND HEIGHT VECTOR COORDINATES
C
       CALL UN2(LOCDA,IV,IH)
       LOC=LOCDA+1
C
C2     RETRIEVE LOCATION OF TRC RADII FOR LOWER BASE AND UPPER BASE
C
       CALL UN2(LOC,IRB,IRTOP)
```

are used to retrieve the locations at which the TRC's vertex, height vector, lower base radius, and upper base radius values are stored in the ASTER array.

The statements

```
C
C3     RETRIEVE COORDINATES OF VERTEX AND HEIGHT VECTOR
C
       V(1)=ASTER(IV)
       V(2)=ASTER(IV+1)
       V(3)=ASTER(IV+2)
       H(1)=ASTER(IH)
       H(2)=ASTER(IH+1)
       H(3)=ASTER(IH+2)
```

are used to retrieve the x, y, z coordinates of the vertex and height vector and store them into two three-element arrays, respectively.

The statements

```
C
C4     RETRIEVE RADII OF LOWER AND UPPER BASES
C
       RB=ASTER(IRB)
       RT=ASTER(IRTOP)
```

are used to retrieve the values of the radii of the lower and upper bases, respectively.

The statements

```
RIN=-PINF
ROUT=PINF
```

are used to initialize the variable RIN to an extremely large negative value and the variable ROUT to an extremely large positive value.

The statements

```
LRO=0
LRI=0
```

are used to initialize the entering surface number variable and the exit surface number variable to zero.

The statements

```
INTSEC=0
INTR1=0
INTR2=0
```

are used to initialize the variables INTSEC, INTR1, and INTR2 to zero.

The statements

```
C
C*    COMPUTE COORDINATES OF (V-XB)
C
      V1XB1=V(1)-XB(1)
      V2XB2=V(2)-XB(2)
      V3XB3=V(3)-XB(3)
```

are used to compute $\overline{V}_x-\overline{XB}_x$, $\overline{V}_y-\overline{XB}_y$, and $\overline{V}_z-\overline{XB}_z$.

The statements

```
C
C*    COMPUTE DOT PRODUCTS
C
      PVPV=V1XB1*V1XB1+V2XB2*V2XB2+V3XB3*V3XB3
```

```
VPW=V1XB1*WB(1)+V2XB2*WB(2)+V3XB3*WB(3)
WH =WB(1)*H(1)+WB(2)*H(2)+WB(3)*H(3)
VPH=V1XB1*H(1)+V2XB2*H(2)+V3XB3*H(3)
HH=H(1)*H(1)+H(2)*H(2)+H(3)*H(3)
```

are used to compute the dot products $(\overline{V}-\overline{XB})\cdot(\overline{V}-\overline{XB})$, $(\overline{V}-\overline{XB})\cdot\overline{WB}$, $\overline{WB}\cdot\overline{H}$, $(\overline{V}-\overline{XB})\cdot\overline{H}$, and $\overline{H}\cdot\overline{H}$, respectively, which in turn are used in solving the quadratic Equation (49).

The statement

```
RTRB=RT-RB
```

is used to compute the difference between the radius of the upper base and the radius of the lower base (R_T-R_B) for use in solving quadratic Equation (49).

The statement

```
C
C7      COMPUTE C2 QUANTITY OF QUADRATIC EQUATION
C
        RBRTVP=RB-VPH*RTRB/HH
```

is used to compute the C_1 and C_2 portion of the constant term from quadratic Equation (49).

The statement

```
VPHHH=VPH+HH
```

is used to compute the value of $(\overline{V}-\overline{XB})\cdot\overline{H} + \overline{H}\cdot\overline{H}$ for solving RIN or ROUT with planar surface $\overline{V}+\overline{H}$.

The statements

```
UM=HH*(PVPV-RBRTVP**2)-VPH*VPH
AMBD=HH*VPW-WH*(VPH-RTRB*RBRTVP)
DEN=HH-WH**2*(1.+RTRB**2/HH)
```

are used to compute the values of the constant term μ', the coefficient of 2S or λ', and the coefficient of S^2 or τ for solving quadratic Equation (49).

$$\boxed{\begin{aligned} & S^2 \left\{ 1 - \frac{(\overline{WB}\cdot\overline{H})^2 \left[1.0 + (R_T - R_B)^2\right]}{\overline{H}\cdot\overline{H}} \right\} \\ & - 2S \left\{ - (\overline{XB} - \overline{V})\cdot\overline{WB} + \frac{(\overline{WB}\cdot\overline{H})\cdot\left[(\overline{XB} - \overline{V})\cdot\overline{H} + C_1\right]}{\overline{H}\cdot\overline{H}} \right\} \\ & + \left\{ \left[(\overline{XB} - \overline{V})^2 - C_2^2\right] - \frac{\left[(\overline{XB} - \overline{V})\cdot\overline{H}\right]^2}{\overline{H}\cdot\overline{H}} \right\} = 0 \end{aligned}} \qquad (49)$$

where

$$C_1 = \left(R_T - R_B\right) \left\{ R_B - \frac{\left(R_T - R_B\right)\left[\left(\overline{V} - \overline{XB}\right)\cdot\overline{H}\right]}{\overline{H}\cdot\overline{H}} \right\}$$

$$C_2 = R_B - \frac{\left(R_T - R_B\right)\left[\left(\overline{V} - \overline{XB}\right)\cdot\overline{H}\right]}{\overline{H}\cdot\overline{H}}$$

Letting τ, λ', and μ be the coefficient of S^2, 2S, and the constant term, respectively, Equation (49) takes the form

$$\tau S^2 - 2\lambda' S + \mu' = 0$$

If

$$\lambda = \lambda'/\tau \text{ and } \mu = \mu'/\tau$$

then

$$S^2 - 2\lambda S + \mu = 0$$

Therefore, the possible intersect points (RIN and ROUT) for the intersection of a ray with the right truncated cone are:

$$\boxed{\text{RIN} = \lambda - \sqrt{\lambda^2 - \mu}} \qquad (51)$$

$$\boxed{\text{ROUT} = \lambda + \sqrt{\lambda^2 - \mu}} \qquad (52)$$

For $\lambda^2 - \mu \leq 0$ no intersection with the cone is possible.

The statement

```
C
C8     TEST FOR RAY PARALLEL TO EITHER SIDE OF CONE
C
       IF(ABS(DEN).GT.1.0E-6)GOTO 40
```

is used to determine if the ray is parallel to either side of the cone, which means that $\tau=0$. If not, control is passed to another section of the subroutine where the roots of the quadratic are computed.

The statement

```
       IF(RTRB.EQ.0.)GOTO 200
```

is used to determine if the radii of the upper and lower bases are equal such that the right truncated cone is actually a right circular cylinder. If this is the case, the ray can intersect only the upper and lower planes of the body, since it was previously determined that the ray is parallel to one of the sides of the body. Control is transferred to the section of the subroutine that computes the intersection with the two planes.

The statement

```
C
C9     COMPUTE INTERSECT WITH QUADRATIC SURFACE FOR RAY PARALLEL TO SIDE
C
       R2=UM/(2.*AMBD)
```

is used to solve quadratic Equation (49) when τ, the coefficient of S^2, is equal to zero. This results in the one possible intersect point of the ray with quadratic surface.

The statement

```
      F1=R2*WH-VPH
```

is used to compute the quantity $S(\overline{WB}\cdot\overline{H})-(\overline{V}-\overline{XB})\cdot\overline{H}$, where S is the value computed in the previous step for the intersect point with the quadratic surface. For this intersect to be valid, that is, to intersect the real part of the cone, the intersect point must be on the cone's surface between vertex $\overline{V}$ and height vector $\overline{H}$ such that

$$\boxed{0 \leq (\overline{XB}-\overline{V})\cdot\overline{H} + \overline{WB}\cdot\overline{H}\cdot S \leq \overline{H}\cdot\overline{H}} \tag{53}$$

where S is the intersect point on the quadratic surface.

The statements

```
C
C10   TEST IF INTERSECT BETWEEN PLANAR SURFACES
C
      IF(F1.LT.0.)GOTO 200
      IF(F1.GT.HH)GOTO 200
```

are used to determine if the intersect point on the quadratic surface satisfies Equation (53); that is, to determine if the intersect point lies between the two planar surfaces.

The statement

```
      INTSEC=INTSEC+1
```

is used to add one to the variable INTSEC, which is used later in the subroutine to determine if all possible intersects have been made.

The statements

```
      IF(WH.LE.0.)GOTO 10
      IF(RTR8)20,20,30
   10 IF(RTR8)30,30,20
```

are used to determine the direction of the ray with respect to height vector $\overline{H}$ and to determine the direction of the point of the cone. This is done to determine if the intersect point on the quadratic surface is an entry point or an exit point.

The statements

```
C
C11   ASSIGN SURFACE EXIT NUMBER AND ROUT FOR QUADRATIC SURFACE
C
   20 LRO=3
      ROUT=R2
      GOTO 250
```

are used to assign the intersect point on the quadratic surface previously determined (ray parallel to either side of cone) to ROUT, since the ray is in the direction of the point of the cone. The surface number for an exit from the quadratic surface is also assigned. Control is then transferred to another section of the subroutine to compute RIN with the larger planar surface on the cone.

The statements

```
C
C12   ASSIGN SURFACE ENTRY NUMBER AND RIN FOR QUADRATIC SURFACE
C
   30 LRI=3
      RIN=R2
      INTSEC=INTSEC+1
      GOTO 210
```

are used to assign the intersect point on the quadratic surface previously determined (ray parallel to either side of cone) to RIN, since the ray is in the opposite direction of the point of the cone. The surface number for an entrance into the quadratic surface is also assigned. Control is transferred to another section of the subroutine to compute ROUT with the larger planar surface on the cone. The variable INTSEC is also advanced by a count of one.

The statements

```
C
   40 AMBDA=AMBD/DEN
      UMU=UM/DEN
      DISC=AMBDA**2-UMU
```

are used to compute the values of λ and μ. The quantity $\lambda^2-\mu$ is computed for solving Equations (51) and (52).

The statement

```
      IF(DISC)350,200,50
```

is used to determine the sign of $\lambda^2-\mu$, the quantity under the radical of Equations (51) and (52). If negative, no intersection with the cone is possible, and control is transferred to another section of the subroutine to record a total miss. If zero, control is transferred to the section of the subroutine to compute possible intersects with the planar surfaces. If positive, intersections with the quadratic surface occurs, and control is transferred to compute those intersections.

The statements

```
C
C13   SOLVE FOR VALUES OF QUADRATIC EQUATION
C
   50 SD=SQRT(DISC)
      R1=AMBDA-SD
      R2=AMBDA+SD
```

are used to compute the quantity $\sqrt{\lambda^2-\mu}$ and to compute possible values of RIN and ROUT with the quadratic surface from Equations (51) and (52).

The statement

```
      F1=R2*WH-VPH
```

is used to compute the quantity $R2(\overline{WB}\cdot\overline{H})-(\overline{V}-\overline{XB})\cdot\overline{H}$, where R2 is one of the possible intersects just computed for the quadratic surface.

The statements

```
C
C14   TEST FOR INTERSECT BETWEEN PLANAR SURFACES
C
      IF(F1.LT.0.)GOTO 60
      IF(F1.LE.HH)INTR2=INTR2+1
```

are used to determine if Equation (53) is satisfied; that is, if the R2 intersect from Equation (52) for the quadratic surface lies on the cone's surface between the two planar surfaces. If the equation is satisfied, the variable INTR2 is advanced by a count of one to indicate that a valid intersect for R2 has been found.

The statement

```
   60 F1=R1*WH-VPH
```

is used to compute the quantity $R1(\overline{WB}\cdot\overline{H})-(\overline{V}-\overline{XB})\cdot\overline{H}$, where R1 is one of the possible intersects just computed for the quadratic surface.

The statements

```
      IF(F1.LT.0.)GOTO 70
      IF(F1.LE.MH)GOTO 80
```

are used to determine if Equation (53) is satisfied; that is, if the R1 intersect from Equation (51) for the quadratic surface lies on the cone's surface between the two planar surfaces.

The statement

```
   70 IF(INTR2.LT.1)GOTO 200
```

is executed if R1 is above or below the TRC. It is used to determine if R2 was a valid intersect on the quadratic surface of the TRC by testing the count in the variable INTR2. If R2 was not valid, control is transferred to determine the intersections with the planar surfaces.

The statements

```
 ROUT=R2
 RIN=R2
 LRO=3
 LRI=3
 INTSEC=INTSEC+1
 GOTO 200
```

are used to set both RIN and ROUT equal to R2 since R2 was a valid intersect, but R1 was not, and the direction of the ray has not yet been determined. The entry and exit surface numbers for a quadratic surface are assigned for the same reason. The variable INTSEC is also advanced by a count of one to record that thus far one valid intersection has been found. Control is then transferred to determine the intersections with the planar surfaces.

The statement

```
80 INTR1=INTR1+1
```

is executed if R1 is a valid intersect on the quadratic surface of the TRC. It is used to indicate that a valid intersect for R1 has been found.

The statement

```
 IF(INTR2.GE.1)GOTO 90
```

is used to determine if R2 was also a valid intersect on the quadratic surface by testing the variable INTR2.

The statements

```
 ROUT=R1
 RIN=R1
 LRO=3
 LRI=3
 INTSEC=INTSEC+1
 GOTO 200
```

are used to set both RIN and ROUT equal to R1 since R1 was determined to be a valid intersect but R2 was not and the direction of the ray has not yet been determined. The entry and exit surface numbers for a quadratic surface are assigned for the same reason. The variable INTSEC is advanced by the count of one to record that thus far one valid intersect has been found. Control is then transferred to determine the intersections with the planar surfaces.

The statement

```
   90 IF(R1-R2)100,350,110
```

is used to determine RIN and ROUT from R1 and R2 when R1 and R2 are both valid intersections on the quadratic surface.

The statements

```
C
C15   COMPUTE RIN AND ROUT FOR QUADRATIC SURFACE
C
  100 RIN=R1
      ROUT=R2
      LRO=3
      LRI=3
      GOTO 300
  110 RIN=R2
      ROUT=R1
      LRO=3
      LRI=3
      GOTO 300
```

are used to assign RIN and ROUT from the values of R1 and R2 depending upon which value of R1 or R2 was the larger as determined by the previous test. The entry and exit surface variables are also assigned values for the quadratic surface since RIN and ROUT both occur on the quadratic surface of the TRC.

The statement

```
C
  200 IF(WH)210,350,250
```

is used to test the results of the dot product $\overline{WB}\cdot\overline{H}$ to determine the direction of the ray with respect to the $\overline{H}$ vector ($\overline{WB}\cdot\overline{H} < 0$ for a downward direction; $\overline{WB}\cdot\overline{H} > 0$ for an upward direction; $\overline{WB}\cdot\overline{H} = 0$ for perpendicular). This statement begins the sections for computing the intersects with the planar surfaces.

The statement

```
210 IF(VPH.GE.0.)GOTO 350
```

is used to determine if the ray (moving in the opposite direction with respect to $\overline{H}$) will intersect the surface plane containing $\overline{V}$. A positive result of the test would indicate a negative distance. Therefore, control is passed to another area of the subroutine where the total miss condition is recorded. A negative result would mean a positive distance; therefore the ray intersects the surface plane containing $\overline{V}$.

The statement

```
CP=VPH/WH
```

is used to compute the intersect of the ray with the plane surface containing $\overline{V}$.

The statements

```
F1=CP*CP-2.*CP*VPW+PVPV-RB*RB
IF(F1.GT.0.)GOTO 220
```

are used to evaluate Equation (57) for S=ROUT to determine if the intersect with the plane surface containing $\overline{V}$ is within the cross-section of the R_B base.

$$\boxed{S^2 - 2S\,[\overline{WB}\cdot(\overline{V}-\overline{XB})] + (\overline{V}-\overline{XB})^2 - R_B{}^2 \leq 0} \tag{57}$$

If Equation (57) is satisfied, a valid intersect occurs. If the equation is not satisfied, control is transferred to determine if there is a valid intersect with the plane surface containing $\overline{V}+\overline{H}$.

The statements

```
C
C16    COMPUTE ROUT FOR EXIT FROM V-PLANE SURFACE
C
       INTSEC=INTSEC+1
       ROUT=CP
       LRO=1
```

are used to advance the count in INTSEC by one as a result of the valid intersect with the $\overline{V}$ plane. ROUT is assigned a value from the previously computed intersect with the $\overline{V}$ plane, and the surface number variable is assigned for an exit from the $\overline{V}$ plane.

The statement

```
       IF(INTSEC.GE.2)GOTO 300
```

is used to determine if two valid intersects have been found. If true, control is passed to the end of the subroutine. If not, the intersect with the $\overline{V}+\overline{H}$ plane will be determined.

The statement

```
   220 CM=VPHHH/WH
```

is used to compute the intersect of the ray with the $\overline{V}+\overline{H}$ plane surface.

The statements

```
       F1=CM*CM-2.*((VPW+WH)*CM-VPH)+HH+PVPV-RT*RT
       IF(F1.GT.0.)GOTO 350
```

are used to evaluate Equation (60) for S=RIN to determine if the intersect with the $\overline{V}+\overline{H}$ plane surface is within the cross-section of the R_T base.

$$\boxed{S^2-2S\ [\overline{WB}\cdot(\overline{V}-\overline{XB}) + (\overline{WB}\cdot\overline{H})] + (\overline{V}-\overline{XB})^2 + 2(\overline{V}-\overline{XB})\cdot\overline{H} + \overline{H}\cdot\overline{H} - R_T^{\ 2} \leq 0} \qquad (60)$$

If Equation (60) is satisfied, a valid intersect occurs. If the equation is not satisfied, control is transferred to the end of the subroutine where a total miss is recorded.

The statements

```
C
C17   COMPUTE RIN FOR ENTRY INTO V*H PLANE SURFACE
C
      RIN=CM
      LRI=2
      GOTO 300
```

are used to assign to RIN the value just computed for an intersect with the $\overline{V}+\overline{H}$ plane, and to assign the surface number variable for an entry into the $\overline{V}+\overline{H}$ plane. Control is then transferred to the end of the program for a final check of RIN and ROUT.

The statement

```
  250 IF(VPHHH.LT.0.)GOTO 350
```

is used to determine if the ray (moving in a direction within 90° of vector $\overline{H}$) will intersect the $\overline{V}+\overline{H}$ surface plane. A negative result of the test would indicate a negative distance. Therefore, control is passed to another area of the subroutine where the total miss condition is recorded. A positive result indicates a positive distance; therefore the ray intersects the $\overline{V}+\overline{H}$ surface plane.

The statement

```
      CP=VPHHH/WH
```

is used to compute the intersect of the ray with the $\overline{V}+\overline{H}$ surface plane.

The statements

```
      F1=CP*CP-2.*((VPW+WH)*CP-VPH)+HH+PVPV-RT*RT
      IF(F1.GT.0.)GOTO 260
```

are used to evaluate Equation (60) for S=ROUT to determine if the intersect with the $\overline{V+H}$ plane surface is within the cross-section of the R_T base. If the equation is satisfied, a valid intersect occurs. If the equation is not satisfied, control is transferred to determine if there is a valid intersect with the $\overline{V}$ plane surface.

The statements

```
C
C18   COMPUTE ROUT FOR EXIT FROM V+H PLANE SURFACE
C
      INTSEC=INTSEC+1
      ROUT=CP
      LRO=2
```

are used to advance the count in INTSEC by one as a result of the valid intersect with the $\overline{V+H}$ plane. ROUT is assigned a value from the previously computed intersect with the $\overline{V+H}$ plane, and the surface number variable is assigned for an exit from the $\overline{V}$ plane.

The statement

```
  260 IF(INTSEC.GE.2)GOTO 300
```

is used to determine if two valid intersects have been found. If so, control is passed to the end of the subroutine for a final test of RIN and ROUT. If not, the intersect with the $\overline{V}$ plane will be determined.

The statement

```
      CM=VPH/WH
```

is used to compute the intersect of the ray with the $\overline{V}$ plane surface.

The statements

```
      F1=CM*CM-2.*CM*VPW+PVPV-RB*RB
      IF(F1.GT.0.)GOTO 350
```

are used to evaluate Equation (57) for S=RIN to determine if the intersect with the $\overline{V}$ plane surface is within the cross-section of the R_B base. If the equation is satisfied, a valid intersect occurs. If the equation is not satisfied, control is transferred to record a total miss.

The statements

```
C
C19    COMPUTE RIN FOR ENTRY INTO V-PLANE SURFACE
C
       RIN=CM
       LRI=1
```

are used to assign RIN the value just computed for an intersect with the $\overline{V}$ plane and to assign the surface number variable for an entry into the $\overline{V}$ plane.

The statement

```
C
  300 IF(ABS(ROUT-RIN)-ROUT*1.0E-5)350,350,360
```

is used to determine if the distance between RIN and ROUT is extremely small. If it is, a miss is recorded for RIN and ROUT. If not, control is returned to the calling program.

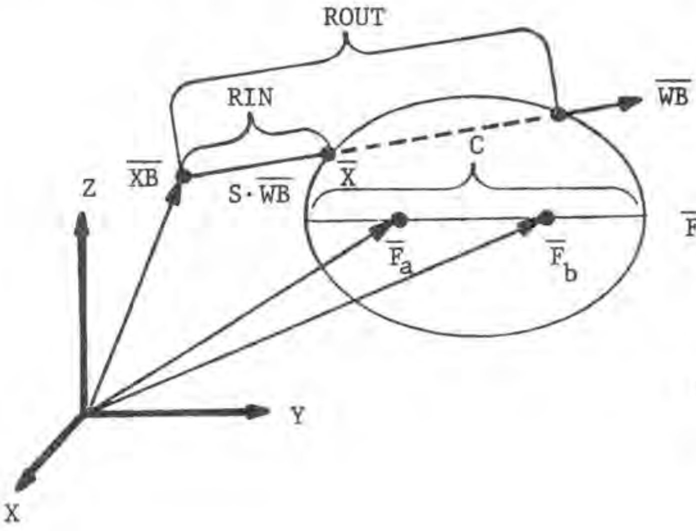
The statements

```
C
C20    RAY MISSES TRC
C
  350 RIN=PINF
      ROUT=-PINF
      LRI=0
      LRO=0
  360 RETURN
```

are used to record a miss of the TRC. RIN is set to an extremely large positive value and ROUT is set to an extremely large negative value. The surface number entry and exit variables are set to zero. Control is then returned to the calling program.

Subroutine ELL

This subroutine computes the ray intersections with an ellipsoid of revolution as illustrated in Figure 66.



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
C	length of the major axis
$\overline{F}_a, \overline{F}_b$	foci of the ellipsoid
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 66. Ray Intersection with Ellipsoid of Revolution

The statements

```
 DIMENSION FOCIA(3),FOCIB(3)
 DIMENSION MASTER(10000)
 COMMON ASTER(10000)
 COMMON/PAREM/XB(3),WB(3),IR
 COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
 COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and pass information into and out of the subroutine.

The statement

```
 EQUIVALENCE (ASTER,MASTER)
```

is used to establish equivalence between the ASTER and MASTER arrays.

The statements

```
C
C1    RETRIEVE LOCATION OF ELLIPSE FOCI AND LENGTH STORAGE POSITIONS
C
      CALL UN2(LOCDA,IV1,IV2)
      IRR=MASTER(LOCDA+1)
```

are used to retrieve the pointers to the storage locations of the ellipse foci and length of the major axis.

The statements

```
      FOCIA(1)=ASTER(IV1)
      FOCIA(2)=ASTER(IV1+1)
      FOCIA(3)=ASTER(IV1+2)
      FOCIB(1)=ASTER(IV2)
      FOCIB(2)=ASTER(IV2+1)
      FOCIB(3)=ASTER(IV2+2)
      C=ASTER(IRR)
```

are used to retrieve the values of the foci coordinates and major axis length from the ASTER array.

The statements

```
      RIN=PINF
      ROUT=-PINF
```

are used to initialize the value of RIN to an extremely large positive number and ROUT to an extremely large negative number.

The statements

```
C
C2    COMPUTE COORDINATES FOR VECTOR D1 AND D2
C
      D1X=XB(1)-FOCIA(1)
      D1Y=XB(2)-FOCIA(2)
      D1Z=XB(3)-FOCIA(3)
      D2X=XB(1)-FOCIB(1)
      D2Y=XB(2)-FOCIB(2)
      D2Z=XB(3)-FOCIB(3)
```

are used to compute the coordinates of $\overline{D}_1$ and $\overline{D}_2$ using the relationship

$$\overline{D}_x = \overline{XB}_x - \overline{F}_x$$

$$\overline{D}_y = \overline{XB}_y - \overline{F}_y$$

$$\overline{D}_z = \overline{XB}_z - \overline{F}_z$$

The statements

```
C
C3     COMPUTE DOT PRODUCTS
C
       A1=2.*(D1X*WB(1)+D1Y*WB(2)+D1Z*WB(3))
       A2=2.*(D2X*WB(1)+D2Y*WB(2)+D2Z*WB(3))
       B1=D1X*D1X+D1Y*D1Y+D1Z*D1Z
       B2=D2X*D2X+D2Y*D2Y+D2Z*D2Z
```

are used to compute vector dot products required for Equation (64).

$$\boxed{\frac{\overline{D}_1^{\,2} - \overline{D}_2^{\,2} - C^2}{-2C} + \frac{\left(2\overline{D}_1\cdot\overline{WB} - 2\overline{D}_2\cdot\overline{WB}\right)}{-2C}\,S = \sqrt{\left(\overline{D}_2 + \overline{WB}\cdot S\right)^2}} \qquad (64)$$

where:

$$A1 = 2\overline{D}_1 \cdot \overline{WB}$$

$$A2 = 2\overline{D}_2 \cdot \overline{WB}$$

$$B1 = \overline{D}_1^{\,2}$$

$$B2 = \overline{D}_2^{\,2}$$

The statements

```
C
C4    COMPUTE A AND B
C
      AA=(A2-A1)/(2.*C)
      BB=(C*C+B2-B1)/(2.*C)
```

are used to compute the A and B terms in Equation (64).

$$A = BB = \frac{C^2 + \overline{D}_2{}^2 - \overline{D}_1{}^2}{2C}$$

$$B = AA = \frac{2\overline{D}_2 \cdot \overline{WB} - 2\overline{D}_1 \cdot \overline{WB}}{2C}$$

The statements

```
C
C5    COMPUTE LAMBDA AND MU
C
      ALAMD=AA*AA-1.
      ALAM1=(AA*BB-.5*A2)/ALAMD
      U=(BB*BB-B2)/ALAMD
```

are used to compute λ_1 and μ using the equations

$$\boxed{\lambda = \frac{A \cdot B - \overline{D}_2{}^2 \cdot \overline{WB}}{B^2-1}} \tag{67}$$

$$\boxed{\mu = \frac{A^2 - \overline{D}_2{}^2}{B^2-1}} \tag{68}$$

The statements

```
C
C6    COMPUTE RIN AND ROUT
C
      DISCRM=ALAM1*ALAM1-U
      IF(DISCRM.LE.0.)RETURN
      SQRTDI=SQRT(DISCRM)
      RIN=-ALAM1-SQRTDI
      ROUT=-ALAM1+SQRTDI
      RETURN
```

are used to compute RIN and ROUT according to the equations

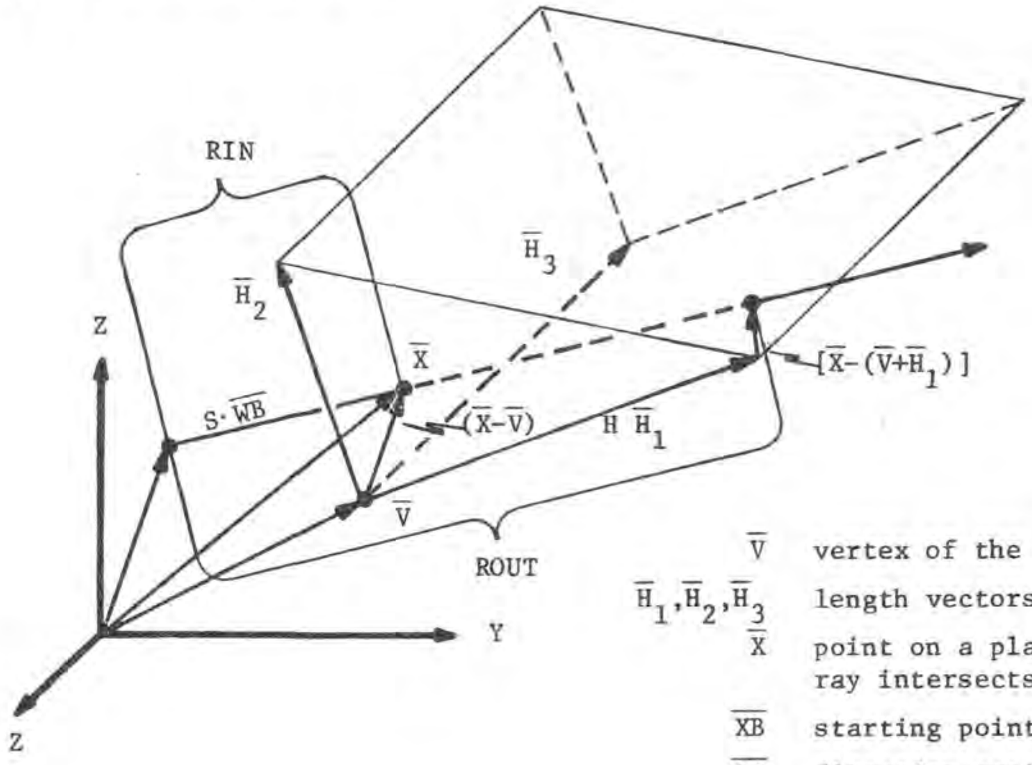
$$\boxed{\text{RIN} = -\lambda - \sqrt{\lambda^2 - \mu}} \qquad (69)$$

$$\boxed{\text{ROUT} = -\lambda + \sqrt{\lambda^2 - \mu}} \qquad (70)$$

If $\lambda^2 - \mu \le 0$, no intersection occurs, and execution returns to the calling routine with RIN set at 10^{50} and ROUT set at -10^{50}.

Subroutine RAW

This subroutine computes the ray intersections with a right angle wedge. The right angle wedge is defined by vertex $\overline{V}$ and length vectors $\overline{H}_1$, $\overline{H}_2$, and $\overline{H}_3$ as shown in Figure 67.



$\overline{V}$	vertex of the RAW
$\overline{H}_1,\overline{H}_2,\overline{H}_3$	length vectors of the RAW
$\overline{X}$	point on a plane where the ray intersects the plane
$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{SWB}$	distance from XB to point where ray intersects plane
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 67. Right Angle Wedge

The statements

```
      DIMENSION H1(3),H2(3),H3(3),V(3),ASQ(3),PV(4),G(3)
      COMMON ASTER(10000)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and pass information into and out of this subroutine.

The statements

```
C
CI    RETRIEVE LOCATIONS OF VERTEX AND LENGTH VECTORS
C
```

```
CALL UN2(LOCDA,IV,IH1)
LOC=LOCDA+1
CALL UN2(LOC,IH2,IH3)
```

are used to retrieve the storage locations of the vertex and length vectors in the ASTER array.

The statements

```
H1(1)=ASTER(IH1)
H1(2)=ASTER(IH1+1)
H1(3)=ASTER(IH1+2)
H2(1)=ASTER(IH2)
H2(2)=ASTER(IH2+1)
H2(3)=ASTER(IH2+2)
H3(1)=ASTER(IH3)
H3(2)=ASTER(IH3+1)
H3(3)=ASTER(IH3+2)
V(1)=ASTER(IV)
V(2)=ASTER(IV+1)
V(3)=ASTER(IV+2)
```

are used to retrieve the x, y, and z coordinates of the vertex and length vectors from the ASTER array.

The statements

```
RIN=-PINF
ROUT=PINF
CM=-PINF
CP=PINF
L=0
L1=0
K=0
LRI=0
LRO=0
```

are used to initialize variables prior to the start of computations.

The statements

```
C
C2     COMPUTE A
C
       ASQ(1)=H1(1)*H1(1)+H1(2)*H1(2)+H1(3)*H1(3)
       ASQ(2)=H2(1)*H2(1)+H2(2)*H2(2)+H2(3)*H2(3)
       ASQ(3)=H3(1)*H3(1)+H3(2)*H3(2)+H3(3)*H3(3)
```

are used to compute the simplifying constant A_i using Equation

$$\boxed{A_i = \overline{H}_i \cdot \overline{H}_i, \quad i = 1,2,3} \tag{72}$$

The statements

```
C
C3     COMPUTE P
C
       XB1V1=XB(1)-V(1)
       XB2V2=XB(2)-V(2)
       XB3V3=XB(3)-V(3)
       PV(1)=XB1V1*H1(1)+XB2V2*H1(2)+XB3V3*H1(3)
       PV(2)=XB1V1*H2(1)+XB2V2*H2(2)+XB3V3*H2(3)
       PV(3)=XB1V1*H3(1)+XB2V2*H3(2)+XB3V3*H3(3)
```

are used to compute the simplifying constant P_i using Equation

$$\boxed{P_i = (\overline{XB}-\overline{V}) \cdot \overline{H}_i, \quad i = 1, 2, 3} \tag{73}$$

The statements

```
C
C4     COMPUTE G
C
```

```
      G(1)=WB(1)*H1(1)+WB(2)*H1(2)+WB(3)*H1(3)
      G(2)=WB(1)*H2(1)+WB(2)*H2(2)+WB(3)*H2(3)
      G(3)=WB(1)*H3(1)+WB(2)*H3(2)+WB(3)*H3(3)
```

are used to compute the simplifying constant G_i using Equation

$$\boxed{G_i = \overline{WB}\cdot\overline{H}_i, \quad i = 1, 2, 3} \tag{74}$$

The statement

```
C
      DO 140 I=1,2
```

is used to start a loop which will test G_1, P_1, G_2, and P_2 to determine if intersections are possible.

The statement

```
      IF(G(I))10,110,60
```

is used to test G_1 and G_2 and branches accordingly.

The statement

```
   10 IF(-PV(I))20,400,400
```

is used to test P_1 or P_2 if G_1 or G_2 is less than zero. If $-P_i \geq 0$ and $G_i < 0$ the ray is moving away, no intersection is possible, and execution transfers to Statement 400.

The statement

```
C
C5    COMPUTE S1 OR S3
C
   20 TEMP=-PV(I)/G(I)
      IF(TEMP-CP)30,130,130
```

are used to compute the intersect distance using Equation (75) if $-P_i < 0$.

$$\boxed{S_3 = -P_1/G_1} \tag{75}$$

or

$$\boxed{S_1 = -P_2/G_2} \tag{76}$$

The computed distance is compared with the previously computed value of CP on the initialized value. If the computed value is greater, execution transfers to Statement 130.

The statements

```
30 CP=TEMP
   L=I
   GOTO(40,50),I
40 LRO=3
   GOTO 130
50 LRO=1
   GOTO 130
```

are used to set CP equal to the computed value and LRO equal to three for I = 1 and LRO equal to one for I = 2 if the computed value is less. Execution then transfers to Statement 130.

The statement

```
60 IF(-PV(I).LE.0.)GOTO 130
```

is used to test P_1 or P_2 if G_1 or G_2 is greater than zero. If $-P_i \leq 0$ an intersection with the face in question will not occur, and execution transfers to Statement 130.

The statements

```
C
C6    COMPUTE S1 OR S3
C
      TEMP=-PV(I)/G(I)
      IF(TEMP.LE.CM)GOTO 130
```

are used to compute the intersect distance using Equation (75) or (76) if $-P_i > 0$. The computed distance is compared with the previously computed value of CM or the initialized value. If the computed value is less than or equal to CM, execution transfers to Statement 130.

The statements

```
      CM=TEMP
      K=I
      GOTO(90,100),I
   90 LRI=3
      GOTO 130
  100 LRI=1
      GOTO 130
```

are used to update CM to the value just computed and set LRI equal to three for I = 1 and LRI equal to one for I = 2 if the computed value is greater than CM. Execution then transfers to Statement 130.

The statements

```
C
  110 IF(PV(I).LE.0.)GOTO 810
      IF(PV(I).GE.ASQ(I))GOTO 810
```

are used to test whether an intersection is possible with the body if G_1 or G_2 is equal to zero. If $P_i \leq 0$ or $P_i \geq A_i$ an intersection is not possible, and the routine is exited via Statement 810.

The statements

```
  130 L1=L1+I
  140 CONTINUE
```

are used to complete the loop computing with G_1, G_2, P_1, and P_2.

The statement

```
C
      IF(G(3))150,210,230
```

is used to test the value of G_3 and branch accordingly.

The statements

```
C
C7    COMPUTE S6
C
  150 TEMP=ASQ(3)-PV(3)
      IF(TEMP.GE.0.)GOTO 180
      TEMP=TEMP/G(3)
      IF(TEMP.LE.CM)GOTO 190
      CM=TEMP
      K=3
      LRI=6
```

are used to compute the intersect distance using Equation (78) if G_3 is less than zero.

$$\boxed{S_6 = (-P_3+A_3)/G_3} \tag{78}$$

If $A_3-P_3 \geq 0$, execution transfers to Statement 180. If $A_3-P_3 < 0$, S_6 is computed and compared to the previously computed value or initial value of CM. If $S_6 \leq$ CM, execution transfers to Statement 190. If $S_6 >$ CM, CM is updated with the value of S_6 and LRI is set to six.

The statement

```
  180 IF(-PV(3))190,400,400
```

is executed if $A_3-P_3 \geq 0$. If $-P_3 \geq 0$ an intersection will not occur, and execution transfers to Statement 400.

The statements

```
C
C8    COMPUTE S5
C
  190 TEMP=-PV(3)/G(3)
      IF(TEMP.GE.CP)GOTO 290
      CP=TEMP
      L=3
      LRO=5
      GOTO 290
```

are used to compute the intersect distance using Equation (77) if $-P_3 < 0$.

$$\boxed{S_5 = -P_3/G_3} \tag{77}$$

If S_5 is greater than or equal to the previously computed value or the initial value of CP, execution transfers to Statement 290. Otherwise, CP is updated with the value of S_5, LRO is set to five, and execution transfers to Statement 290.

The statements

```
C
  210 IF(PV(3).LE.0.)GOTO 400
      IF(PV(3)-ASQ(3))290,290,400
```

are used to test the value of the numerator of Equations (77) and (78) if G_3 is equal to zero. If $P_3 \leq 0$ or $(P_3-A_3) > 0$ an intersection cannot occur, and execution transfers to Statement 400. If $(P_3-A_3) \leq 0$, execution transfers to Statement 290.

The statements

```
C
C9    COMPUTE S5
C
  230 IF(-PV(3).LE.0.)GOTO 260
      TEMP=-PV(3)/G(3)
      IF(TEMP.LE.CM)GOTO 260
      CM=TEMP
      K=3
      LRI=5
```

are used to compute the intersect distance using Equation (77) if G_3 is greater than zero. If $-P_3 \leq 0$, execution transfers to Statement 260. If $-P_3 > 0$, S_5 is computed and its value compared with a previously computed value or the initial value of CM. If S_5 is > CM, CM is updated with the value of S_5 and LRI is set to five.

The statements

```
C
C10    COMPUTE S6
C
  260  TEMP=ASQ(3)-PV(3)
       IF(TEMP.LE.0.)GOTO 400
       TEMP=TEMP/G(3)
       IF(TEMP.GE.CP)GOTO 290
       CP=TEMP
       L=3
       LRO=6
```

are used to compute the intersect distance when $-P_3 \leq 0$. If $(A_3-P_3) \leq 0$, an intersection does not exist, and execution transfers to Statement 400. If $(A_3-P_3) > 0$, the S_6 distance is computed using Equation (78). If S_6 is less than a previously computed value for CP, CP is updated with S_6 and LRO is set to six.

The statements

```
C
C11    COMPUTE S2
C
  290  AG=ASQ(2)*G(1)+ASQ(1)*G(2)
       PV(4)=PV(1)*ASQ(2)+PV(2)*ASQ(1)
       TOP=ASQ(1)*ASQ(2)-PV(4)
       IF(AG)310,350,330
```

are used to compute the terms used by Equation (79).

$$\boxed{S_2 = \left(A_1 \cdot A_2 - P_1 A_2 - P_2 A_1\right) \Big/ \left(A_2 G_1 + A_1 G_2\right)} \qquad (79)$$

A test is made on the value of the denominator, and branching is directed as required. If the denominator is negative, the statements

```
  310  TEMP=TOP/AG
       IF(TEMP.LE.CM)GOTO 380
```

```
      CM=TEMP
      K=4
      LRI=2
      GOTO 380
```

compute the intersect distance S_2 and compare the value with the current value of CM. If S_2 is greater than CM, CM is updated with S_2, LRI is set equal to two, and execution transfers to Statement 380.

The statements

```
C
  330 IF(TOP.LT.0.)GOTO 400
      TEMP=TOP/AG
      IF(TEMP-CP)370,380,380
```

are used to test the value of the numerator if the denominator of Equation (79) is positive. If the numerator is negative, execution transfers to Statement 400. Otherwise, S_2 is computed and compared with CP.

The statements

```
C
  350 IF(PV(4).LE.0.)GOTO 400
      IF(-TOP)380,400,400
```

are used to make additional tests if the denominator is zero. If $(P_1A_2 + P_2A_1) \leq 0$, execution transfers to Statement 400. If the numerator is negative, execution transfers to Statement 400.

The statements

```
  370 CP=TEMP
      L=4
      LRO=2
```

are used to update CP with the value computed for S_2 and set LRO equal to two.

The statements

```
  380 IF(L+K.LE.0)GOTO 400
      ROUT=CP
      RIN=CM
```

are used to check whether CP or CM have been updated with a computed intersection distance. If either has, ROUT is set to the latest value of CP, and RIN is set to the latest value of CM.

The statements

```
C
  400 IF(ROUT.GE.PINF)GOTO 810
      IF(ROUT.LE.0.)GOTO 810
      IF(RIN.GE.ROUT)GOTO 810
      IF(ABS(RIN-ROUT).GT.ROUT*1.0E-5)GOTO 820
```

are used to evaluate the values of RIN and ROUT which have been set in the routine. If ROUT has a positive value less than 10^{50}, and RIN is less than ROUT, an intersection has occurred and the routine is exited via Statement 820 with the computed values of RIN and ROUT.

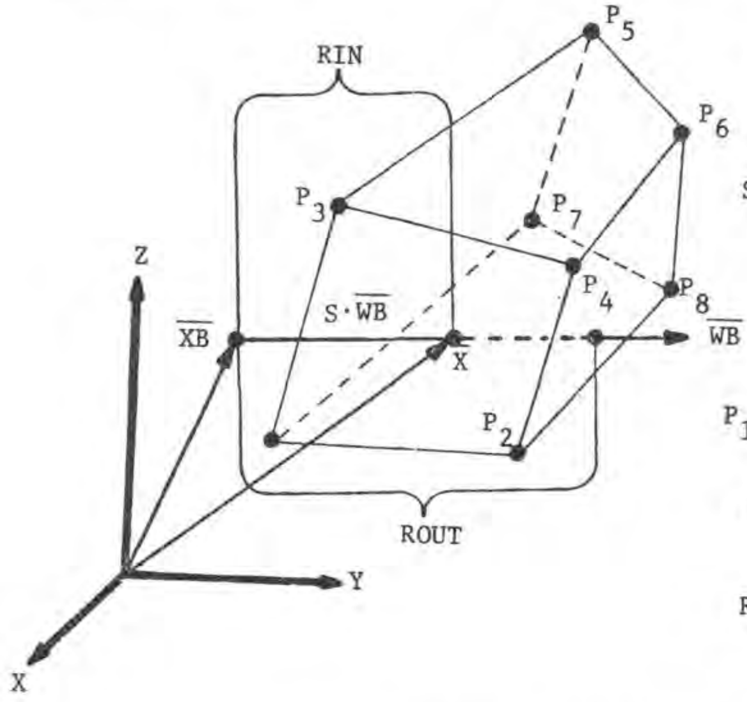
The statements

```
C
  810 ROUT=-PINF
      RIN=PINF
      LRO=0
      LRI=0
  820 RETURN
```

are used to set ROUT equal to -10^{50} and RIN equal to 10^{50}. An intersection is not recorded by LRO or LRI, and execution returns to the calling routine.

Subroutine ARB

This subroutine computes the intersection of a ray with an Arbitrary Polyhedron (ARB). The ARB is described by eight points which define six planar surfaces as shown in Figure 68 below:



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$S\cdot\overline{WB}$	distance from $\overline{XB}$ to point where ray intersects plane
$\overline{X}$	point on plane where ray intersects plane
P_1-P_8	eight points which describe the arbitrary polyhedron
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 68. Arbitrary Polyhedron

An input processing subroutine, Subroutine ALBERT, converts the point and side descriptions (which are input) into six plane equations and adjusts the sign of the coefficients such that

$$\boxed{A_i X + B_i Y + C_i Z + D_i \geq 0 \quad (i = 1,6)} \tag{81}$$

Subroutine ALBERT then stores these data in the ASTER array.

The statements

```
 DIMENSION AA(6,4),XP(3)
 COMMON ASTER(10000)
 COMMON/PAREM/XB(3),WB(3),IR
 COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
 COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and pass information into and out of this routine.

The statements

```
C
C     RETRIEVE PLANAR EQUATIONS FROM ASTER ARRAY
C
      LOC=LOCDA-1
      DO 10 I=1,6
      LOC=LOC+1
      CALL UN2(LOC,LD,LC)
      AA(I,1)=ASTER(LC)
      AA(I,2)=ASTER(LC+1)
      AA(I,3)=ASTER(LC+2)
      AA(I,4)=ASTER(LD)
   10 CONTINUE
```

are used to retrieve the coefficients of the six planar equations stored in the ASTER array. Subroutine UN2 is used to retrieve the locations at which they are stored. The retrieved values are stored in the AA array as follows:

$$AA(I,1) = A_i$$
$$AA(I,2) = B_i$$
$$AA(I,3) = C_i$$
$$AA(I,4) = D_i$$

where I and i = 1, 6.

The statements

```
      RIN=-PINF
      ROUT=PINF
      LRO=0
      LRI=0
      S1=0.
      S2=0.
      L1=0
      L2=0
```

are used to initialize variables which will be utilized during the computations. RIN is set equal to -10^{50}, and ROUT is set equal to 10^{50}. The other variables are set equal to zero.

The statement

```
        DO 70 I=1,6
```

is used to start a loop which performs computations for each face of the ARB.

The statements

```
C
C2      COMPUTE NUMERATOR AND DENOMINATOR OF DISTANCE EQUATION
C
        D=AA(I,4)
        SNUM=-D-AA(I,1)*XB(1)-AA(I,2)*XB(2)-AA(I,3)*XB(3)
        SDEN=AA(I,1)*WB(1)+AA(I,2)*WB(2)+ AA(I,3)*WB(3)
```

are used to compute the numerator and denominator of Equation

$$\boxed{S_i = \frac{-\left(A_i\overline{XB}_x + B_i\overline{XB}_y + C_i\overline{XB}_z + D_i\right)}{A_i\overline{WB}_x + B_i\overline{WB}_y + C_i\overline{WB}_z}} \tag{82}$$

The statements

```
        IF(SDEN)20,70,30
     20 IF(SNUM)40,70,70
     30 IF(SNUM)70,70,40
```

are used to test whether an intersection with the plane is possible. If the denominator is zero, the ray is parallel with the plane and will not intersect. If the numerator and denominator are both greater than zero or both less than zero the ray will intersect the plane.

The statements

```
C
C3      COMPUTE INTERSECT DISTANCE
C
     40 S=SNUM/SDEN
        DO 50 K=1,3
        XP(K)=XB(K)+S*WB(K)
     50 CONTINUE
```

are used to compute the distance from $\overline{XB}$ to the plane in question using Equation (82) and the x, y, and z coordinates of the point at intersection $\overline{XP}$ using the Equation

$$\boxed{\overline{XI}_i = \overline{XB} + \overline{WB} \cdot S_i} \qquad (83)$$

The statements

```
C
C4      TEST IF INTERSECT POINT IS ON ARB
C
        DO 60 J=1,6
        IF(I.EQ.J)GOTO 60
        T=AA(J,1)*XP(1)+AA(J,2)*XP(2)+AA(J,3)*XP(3)+AA(J,4)
        IF(ABS(T).LE.1.0E-6)T=0.
        IF(T.LT.0.)GOTO 70
     60 CONTINUE
```

are used to test whether the intersect point is on the face of the ARB. If the point $\overline{XP}$ is on the face of the ARB, it will be on the (+) side of the five remaining planes. Each of the remaining planes is tested using the relationship

$$\boxed{A_jX + B_jY + C_jZ + D_j \geq 0 \quad (j = 1, 6)} \qquad (84)$$

If the result is less than zero for any of the remaining planes, the ray does not intersect the face of interest.

The statements

```
        IF(L1.GT.0)GOTO 65
        L1=I
        S1=S
        GOTO 70
     65 IF(ABS(S1-S).GT.1.0E-6)GOTO 100
```

are executed if the test described above is satisfied. When the first intersection is computed, L1 is set equal to the face number and S1 is set equal to the computed intersect distance. When the second intersection is computed, the two intersect distances are compared. If

$$|S1-S| \geq 1X10^{-6}$$

execution transfers to Statement 100.

The statement

```
  70 CONTINUE
```

is used to complete the loop considering all six faces.

The statement

```
C
      IF(L1)200,200,150
```

is used to test whether a single intersection has been computed. If no intersections were found, execution is transferred to Statement 200. If one intersection was found, execution transfers to Statement 150.

The statements

```
  100 S2=S
      L2=I
      IF(ABS(S1-S2).LE.S1*1.0E-5)GOTO 200
```

are executed when two intersections are found. S2 is set equal to the second intersection distance, and L2 is set to the face number which contains the intersection. A test is made to insure that S1 and S2 are not the same value within a tolerance of $1X10^{-5}$. If S1 equals S2, the intersections occur at the intersection of two faces, this is assumed to be a miss, and execution transfers to Statement 200.

The statement

```
      IF(S1-S2)110,200,120
```

is used to compare S1 with S2 and branch accordingly. If S1 = S2, execution transfers to Statement 200, which records a miss. If S1 > S2, execution transfers to Statement 120.

The statements

```
110 RIN=S1
    ROUT=S2
    LRI=L1
    LRO=L2
    RETURN
```

are executed if S1 < S2. These are used to set RIN equal to S1, ROUT equal to S2, LRI equal to L1, and LRO equal to L2. Execution then returns to the calling routine.

The statements

```
120 RIN=S2
    LRI=L2
130 ROUT=S1
    LRO=L1
    RETURN
```

are executed if S1 > S2. These are used to set RIN equal to S2, ROUT equal to S1, LRI equal to L2, and LRO equal to L1. Execution then returns to the calling routine.

The statements

```
150 DO 160 J=1,6
    IF(L1.EQ.J)GOTO 160
    T1=AA(J,1)*XB(1)+AA(J,2)*XB(2)+AA(J,3)*XB(3)+AA(J,4)
    IF(ABS(T1).LE.1.0E-6)T1=0.
    IF(T1.LT.0.)GOTO 200
160 CONTINUE
    GOTO 130
```

are executed if only one intersection is obtained, i.e., the ray originated within the ARB.

If the ray originated within the ARB, $\overline{XB}$ will be on the (+) side of each of the six faces. The relationship

$$\boxed{A_iX + B_iY + C_iZ + D_i \geq 0 \quad (i = 1, 6)} \tag{81}$$

is tested. If it is greater than zero for each of the faces, execution transfers to Statement 130, setting ROUT equal to S1 and LRO equal to the face number containing the intersection point. Execution then returns to the calling routine.

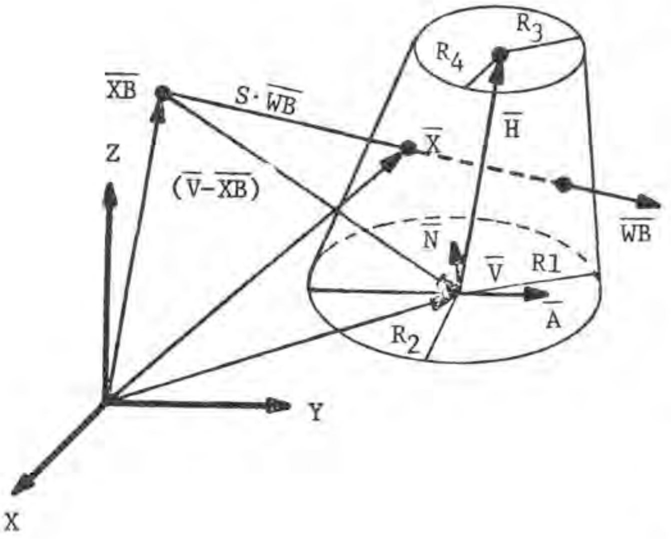
The statements

```
C
  200 RIN=PINF
      ROUT=-PINF
      LRI=0
      LRO=0
      RETURN
      END
C
C
```

are executed if no intersections with the ARB are found. RIN is set to 10^{50}, ROUT is set to -10^{50}, and LRI and LRO are set to zero. Execution then returns to the calling routine.

Subroutine TEC

This subroutine is used to compute the intersection of a ray with a truncated elliptic cone as depicted in the following figure:



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{V}$	vertex of TEC
$\overline{H}$	height vector of TEC
$\overline{N}$	direction of normal to base
$\overline{A}$	direction of semi-major axis
R1	the radius along the semi-major axis of the larger ellipse
R2	the radius along the semi-minor axis of the larger ellipse
R3	the radius along the semi-major axis of the smaller ellipse
R4	the radius along the semi-minor axis of the smaller ellipse

FIG. 69. Truncated Elliptic Cone

The statements

```
      DIMENSION VXB(3),H(3),HN(3),AA(3),BB(3)
      DIMENSION MASTER(10000)
      COMMON ASTER(10000)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGO,
     1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
```

are used to dimension arrays and pass information into and out of this subroutine.

The statement

```
EQUIVALENCE(ASTER,MASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statements

```
C
C1      RETRIEVE LOCATION OF VERTEX AND HEIGHT VECTOR COORDINATES
C
        CALL UN2(LOCDA,IV,IH)
        LOC=LOCDA+1
C
C2      RETRIEVE LOCATION OF NORMAL AND AXES COORDINATES
C
        CALL UN2(LOC,IN,IA)
        LOC=LOC+1
C
C3      RETRIEVE LOCATION OF LENGTHS OF SEMI-MAJOR AXIS AND
C       SEMI-MINOR AXIS OF BASE ELLIPSE
C
        CALL UN2(LOC,IR1,IR2)
```

are used to retrieve the pointers to the coordinates of the vertex, the height vector, the normal, the semi-major axis, and the pointers to the length of the semi-major axis and semi-minor axis of the larger ellipse from the MASTER array.

The statement

```
C
C4      RETRIEVE LOCATION OF THE RATIO OF THE LARGER TO SMALLER ELLIPSE
C
        IR3=MASTER(LOC+1)
```

is used to retrieve the location of the ratio of the larger to the smaller ellipse from the MASTER array.

The statements

```
C
C5     RETRIEVE COORDINATES OF VERTEX AND COMPUTE COORDINATES
C      OF (V-XB) VECTOR
C
       VXB(1)=ASTER(IV)-XB(1)
       VXB(2)=ASTER(IV+1)-XB(2)
       VXB(3)=ASTER(IV+2)-XB(3)
```

are used to retrieve from the ASTER array the x, y, and z components of the vertex and to fill a three-element array with V_x-XB_x, V_y-XB_y, and V_z-XB_z.

The statements

```
C
C6     RETRIEVE COORDINATES OF HEIGHT VECTOR
C
       H(1)=ASTER(IH)
       H(2)=ASTER(IH+1)
       H(3)=ASTER(IH+2)
```

are used to retrieve the x, y, and z components of the height vector from the ASTER array and to store them in a three-element array.

The statements

```
C
C7     RETRIEVE COORDINATES OF NORMAL TO BASE ELLIPSE
C
       HN(1)=ASTER(IN)
       HN(2)=ASTER(IN+1)
       HN(3)=ASTER(IN+2)
```

are used to retrieve the x, y, and z components of the direction of the normal vector from the ASTER array and to store them in a three-element array.

The statements

```
C
C8     RETRIEVE COORDINATES OF SEMI-MAJOR AXIS OF BASE ELLIPSE
C
       AA(1)=ASTER(IA)
       AA(2)=ASTER(IA+1)
       AA(3)=ASTER(IA+2)
```

are used to retrieve the x, y, and z components of the direction of the semi-major axis vector from the ASTER array and to store them in a three-element array.

The statement

```
C
C9      COMPUTE SEMI-MINOR AXIS UNIT VECTOR OF BASE ELLIPSE
C
        CALL CROSS(BB,AA,HN)
```

is used to compute the cross product of the direction of the semi-major axis vector and the direction of the normal vector, which results in the direction of the semi-minor axis.

The statements

```
C
C10     RETRIEVE LENGTHS OF SEMI-MAJOR AND SEMI-MINOR AXES OF BASE
C       ELLIPSE AND RATIO OF LARGER TO SMALLER ELLIPSE
C
        R1=ASTER(IR1)
        R2=ASTER(IR2)
        RR=ASTER(IR3)
```

are used to retrieve the length of the semi-major axis of the base ellipse, the length of the semi-minor axis of the base ellipse, and the ratio of the base to the top ellipse.

The statements

```
C
C11     COMPUTE LENGTHS OF SEMI-MAJOR AND SEMI-MINOR AXES OF TOP ELLIPSE
C
        R3=R1/RR
        R4=R2/RR
```

are used to compute the lengths of the semi-major axis and the semi-minor axis of the top ellipse.

The statements

```
C
C12   START OF COMPUTATIONS FOR DOT PRODUCTS
C
      HDN=DOT(H,HN)
      HDA=DOT(H,AA)
      HDB=DOT(H,BB)
```

are used to compute the dot products of the height vector and the normal vector, the height vector and the semi-major axis vector, and the height vector and the semi-minor axis vector.

The statements

```
      WDN=DOT(WB,HN)
      WDA=DOT(WB,AA)
      WDB=DOT(WB,BB)
```

are used to compute the dot products of the direction cosines of the ray and the normal vector, the direction cosines of the ray and the semi-major axis vector, and the direction cosines of the ray and the semi-minor axis vector.

The statements

```
      VXBDN=DOT(VXB,HN)
      VXBDA=DOT(VXB,AA)
      VXBDB=DOT(VXB,BB)
```

are used to compute the dot products of the $(\overline{V}-\overline{XB})$ vector and the normal vector, the $(\overline{V}-\overline{XB})$ vector and the semi-major axis vector, and the $(\overline{V}-\overline{XB})$ vector and the semi-minor axis vector.

The statement

```
C
C13   TEST TO DETERMINE IF RAY IS PARALLEL TO TOP AND BASE PLANES
C
      IF(ABS(WDN).GT.0.0001)GOTO 20
```

is used to test for the ray parallel to the top and base planes by evaluating the absolute value of the dot product of the direction cosines of the ray and the normal vector $(\overline{WB}\cdot\overline{N})$. For $\overline{WB}\cdot\overline{N} = 0$, the ray is perpendicular to the normal vector and parallel to the top and base planes.

The statement

```
C
C14    COMPUTE RATIO ON NORMAL TO HEIGHT OF HIT
C
       GAMMA = -VXBDN/HDN
```

begins the calculations for determining the intersections of a ray parallel to the top and base planes. This statement computes the ratio on the normal of the height of the hit.

The statement

```
       IF(GAMMA.LT.0.0  .OR.  GAMMA.GT.1.0) GOTO 500
```

is used to test the ratio on the normal of the height of the hit. If the ratio is greater than one or less than zero, the ray is either above or below the body, and a miss occurs.

The statements

```
       A=GAMMA*R3+R1*(1.-GAMMA)
       B=GAMMA*R4+R2*(1.-GAMMA)
```

are used to compute the length of the semi-major axis of the intersection ellipse from Equation (86) and to compute the length of the semi-minor axis of the intersection ellipse from Equation (87).

$$\boxed{M = \gamma R3 + R1\ (1-\gamma)} \qquad (86)$$

$$\boxed{m = \gamma R4 + R2\ (1-\gamma)} \qquad (87)$$

where $\gamma = -\dfrac{(\overline{V}-\overline{XB})\cdot\overline{N}}{\overline{H}\cdot\overline{N}}$

The statements

```
ASQ=A*A
BSQ=B*B
```

are used to compute M squared and m squared of Equations (86) and (87), which will be used to solve Equation (92) for a ray parallel to the top and base planes intersecting the TEC.

$$\boxed{\begin{aligned}&S^2\left[m^2\left(\overline{WB}\cdot\overline{A}\right)^2 + M^2\left(\overline{WB}\cdot\left(\overline{A}\times\overline{N}\right)\right)^2\right]\\ &-2S\left[m^2\left(\overline{WB}\cdot\overline{A}\right)\left(\overline{V}-\overline{XB} + \gamma\overline{H}\right)\cdot\overline{A} + M^2\cdot\overline{WB}\cdot\left(\overline{A}\times\overline{N}\right)\left(\left(\overline{V}-\overline{XB} + \gamma\overline{H}\right)\cdot\left(\overline{A}\times\overline{N}\right)\right)\right]\\ &+m^2\left(\left(\overline{XB}-\overline{V}-\gamma\overline{H}\right)\cdot\overline{A}\right)^2 + M^2\left(\left(\overline{XB}-\overline{V}-\gamma\overline{H}\right)\cdot\left(\overline{A}\times\overline{N}\right)\right)^2 - M^2m^2 = 0\end{aligned}} \qquad (92)$$

which is of the form

$$\tau S^2 - 2S\lambda + \mu = 0$$

Solving for S gives

$$S = \frac{\lambda \pm \sqrt{\lambda \quad - \mu\tau}}{\tau}$$

Therefore

$$\boxed{RIN = \frac{\lambda - \sqrt{\lambda^2 - \mu\tau}}{\tau}} \qquad (93)$$

$$\boxed{ROUT = \frac{\lambda + \sqrt{\lambda^2 - \mu\tau}}{\tau}} \qquad (94)$$

The statements

```
TA=VXBDA+GAMMA*WDA
TB=VXBDB+GAMMA*WDB
```

are used to compute quantities that are used in solving for λ and μ for Equation (92).

The statement

```
DEN=BSQ*WDA*WDA+ASQ*WDB*WDB
```

is used to solve for τ in Equation (92).

The statement

```
IF(ABS(DEN).LE.0.0001)GOTO 500
```

is used to determine if the absolute value of τ of Equation (92) is less than or equal to a very small value, which indicates that the ray misses the TEC.

The statements

```
AMBDA=BSQ*WDA*TA+ASQ*WDB*TB
UM=BSQ*TA*TA+ASQ*TB*TB-ASQ*BSQ
```

are used to compute the values of λ and μ in Equation (92).

The statement

```
DISC=AMBDA*AMBDA-DEN*UM
```

is used to compute the value under the radical of Equations (93) and (94).

The statement

```
IF(DISC.LT.0.0) GOTO 500
```

is used to determine if the value under the radical of Equations (93) and (94) is less than zero, which means that the ray does not intersect the TEC.

The statement

```
    DISC=SQRT(DISC)
```

is used to compute the value under the radical of Equations (93) and (94) if the ray intersects the TEC.

The statements

```
C
C15   COMPUTE RIN AND ROUT AND ASSIGN SURFACE NUMBER FOR
C     RIN AND ROUT WITH QUADRATIC SURFACE
C
      RIN=(AMBDA-DISC)/DEN
      ROUT=(AMBDA+DISC)/DEN
      LRI=3
      LRO=3
      GOTO 400
```

are used to compute the values of RIN and ROUT by solving Equations (93) and (94). The surface number entry and exit variables are also assigned for a RIN and ROUT with the quadratic surface. The subroutine branches to another section to verify the intersection points.

The statement

```
C
C16   SOLVE FOR TERMS IN QUADRATIC EQUATION
C
   20 TAU=(R1/R2)**2
```

begins the calculations for determining the intersection of a ray with the TEC where the ray is not parallel to the top and base planes. This statement is used to compute the square of the ratio of the length of the semi-major axis of the base ellipse to the length of the semi-minor axis of the base ellipse. This will be used to solve Equation (101) for a ray that intersects the TEC and is not parallel to the top and base planes.

$$\boxed{\begin{aligned}&\theta^2\ \Big[\left(\alpha\overline{WB}\cdot\overline{A}-\overline{H}\cdot\overline{A}\right)^2+\left(R1/R2\right)^2\left(\alpha\overline{WB}\cdot\overline{B}-\overline{H}\cdot\overline{B}\right)^2-\left(R1/R2\right)^2\left(R4-R2\right)^2\Big]\\&-2\theta\ \Big[\left(\alpha\overline{WB}\cdot\overline{A}-\overline{H}\cdot\overline{A}\right)\left(\left(\overline{V}-\overline{XB}\right)\cdot\overline{A}-\beta\overline{WB}\cdot\overline{A}\right)+\left(R1/R2\right)^2\left(R2\right)\left(R4\right)-\left(R1/R2\right)^2\left(R2\right)^2\\&+\left(R1/R2\right)^2\left(\alpha\overline{WB}\cdot\overline{B}-\overline{H}\cdot\overline{B}\right)\left(\left(\overline{V}-\overline{XB}\right)\cdot\overline{B}-\beta\overline{WB}\cdot\overline{B}\right)\Big]\\&+\Big[\left(\left(\overline{V}-\overline{XB}\right)\cdot\overline{A}-\beta\overline{WB}\cdot\overline{A}\right)^2+\left(R1/R2\right)^2\left(\left(\overline{V}-\overline{XB}\right)\cdot\overline{B}-\beta\overline{WB}\cdot\overline{B}\right)^2-\left(R1/R2\right)^2\left(R2\right)^2=0\end{aligned}}\qquad(101)$$

where

$$\alpha=\frac{\overline{H}\cdot\overline{N}}{\overline{WB}\cdot\overline{N}}\quad;\quad\beta=\frac{(\overline{V}-\overline{XB})\cdot\overline{N}}{\overline{WB}\cdot\overline{N}}\quad;\quad\delta=(\overline{V}-\overline{XB})$$

which is of the form

$$\tau\theta^2-2\lambda\theta+\mu=0$$

Solving for θ gives

$$\theta=\frac{\lambda}{\tau}\pm\frac{\sqrt{\lambda^2-\tau\mu}}{\tau}$$

where θ is the ratio of the distance from the base plane to the plane of the intersections.

Therefore,

$$\boxed{\theta_1=\frac{\lambda-\sqrt{\lambda^2-\tau\mu}}{\tau}}\qquad(104)$$

$$\boxed{\theta_2=\frac{\lambda+\sqrt{\lambda^2-\tau\mu}}{\tau}}\qquad(105)$$

The equation

$$\boxed{S = \alpha\theta + \beta} \qquad (95)$$

is the equation for computing the distance along the ray to the point of contact where α is the distance along the ray from the plane of the base ellipse to the plane of the top ellipse and β is the distance along the ray from start of ray to the plane of the base ellipse.

Therefore,

$$\boxed{\begin{array}{l} RIN = \alpha\theta_1 + \beta \\ \\ ROUT = \alpha\theta_2 + \beta \end{array}} \qquad (106)$$

where RIN and ROUT are possible candidates for the entry intersection and the exit intersection, respectively.

The statements

```
R2SQ=R2*R2
TR2SQ=TAU*R2SQ
TR4R2=TAU*(R2-R4)**2
TRR4R2=TR2SQ-TAU*R2*R4
```

are used to compute various quantities using the lengths of the semi-major axis of the base and top ellipse and the length of the semi-minor axis of the top ellipse to solve for the coefficients of Equation (101).

The statements

```
BETA=VXBDN/WDN
ALPHA=HDN/WDN
```

are used to compute the distance along the ray from the start point to the plane of the base ellipse and the distance along the ray from the plane of the base ellipse to the plane of the top ellipse.

The statements

```
TA1=ALPHA*WDA-HDA
TB1=ALPHA*WDB-HDB
TA2=VXBDA-BETA*WDA
TB2=VXBDB-BETA*WDB
```

are used to compute quantities for solving τ, λ, and the constant term μ in Equation (101).

The statements

```
DEN   =TA1*TA1+TAU*TB1*TB1-TR4R2
AMBDA=TA1*TA2+TAU*TB1*TB2-TRR4R2
UM    =TA2*TA2+TAU*TB2*TB2-TR2SQ
```

are used to compute the values of τ, λ, and the constant term μ.

The statement

```
IF(ABS(DEN).GT.0.0001)GOTO 150
```

begins the computation for the intersection of the ray and the side of the cone. This statement is used to determine if the absolute value of τ is greater than approximately zero. If not, the ray is parallel to the axis.

The statement

```
IF(R1.EQ.R3)GOTO 100
```

is executed if it was determined by the last instruction that the ray is parallel to the axis. This statement is therefore used to determine if the length of the semi-major axes of the top and base ellipses are equal. This would mean that the sides are parallel to the axis of the TEC; therefore no intersection with the sides would occur for a ray parallel to the axis of the TEC.

The statement

```
IF(AMBDA.NE.0.0)GOTO 110
```

is executed if the sides of the TEC are not parallel to the axis of the TEC as determined by the last instruction. This statement therefore determines if λ from Equation (101) is zero. If it is zero, no intersection with the quadratic surface can take place.

The statements

```
C
C17   THE RAY MISSES THE QUADRATIC SURFACE OF THE TEC
C
  100 S1=-PINF
      S2=PINF
      GOTO 200
```

are used to set temporary values for the intersections with the quadratic surface to an extremely small value and an extremely large value if it was previously determined that the ray could not intersect the quadratic surface of the TEC.

The statement

```
  110 T=UM/(2.*AMBDA)
```

is executed when τ is zero, the sides of the TEC are not parallel, and λ is not zero. Therefore, $\theta = \mu/2\lambda$.

The statement

```
C
C18   COMPUTE DISTANCE TO INTERSECT WITH QUADRATIC SURFACE
C
  120 S=BETA+ALPHA*T
```

is used to compute the distance from the origin of the ray to the intersection on the quadratic surface.

The statement

```
F=S*WON-VXBON
```

is used to compute the vertical distance between the plane of the base ellipse and the plane of the intersection on the quadratic surface.

The statement

```
IF(ABS(F).LE.0.0001)GOTO 125
```

is used to determine if the vertical distance between the plane of the base ellipse and the plane of the intersection on the quadratic surface is very nearly zero. If the distance is approximately zero, the intersection takes place at a corner at the base plane.

The statement

```
IF(F.LT.0.0)GOTO 100
```

is used to determine if the intersection occurs below the plane of the.base ellipse.

The statement

```
IF(ABS(F-HON).LE.0.0001)GOTO 125
```

is used to determine if the vertical distance between the plane of the top ellipse and the plane of the intersection on the quadratic surface is very nearly zero. If the distance is approximately zero, the intersection takes place at a corner at the top plane.

The statement

```
IF(F.GT.HON)GOTO 100
```

is used to determine if the intersection occurs above the plane of the top ellipse.

The statement

```
  125 IF(WDN)130,500,140
```

is used to determine the direction of the ray with respect to the normal. If $\overline{WB} \cdot \overline{N}$ is zero, meaning that the ray is perpendicular to the normal, a miss of the TEC is recorded since execution of this statement meant that the ray was intersecting the quadratic of either the top or base plane.

The statements

```
C
C19   ASSIGN TEMPORARY VALUES TO RIN AND ROUT PER DIRECTION OF RAY
C
  130 S1=S
      S2=PINF
      GOTO 200
  140 S1=-PINF
      S2=S
      GOTO 200
```

are used to store temporary values of RIN and ROUT with the quadratic surface, depending upon the direction of the ray. A branch is then made to compute the possible intersections with the top and base planes.

The statements

```
C
C20   RAY PARALLEL TO SIDE
C
  150 DISC=AMBDA*AMBDA-DEN*UM
      IF(ABS(DISC).GT.0.0001)GOTO 155
      T=AMBDA/DEN
      GOTO 120
```

are used to compute the value of $\lambda^2 - \mu\tau$ and to test the result to determine if the absolute value is large enough for the ray to intersect two sides of the cone. If it is, control is transferred to compute the ray intersections with the side of the cone. If the value is very nearly zero, indicating that the ray is parallel to one side, the value of $\theta = \lambda/\tau$ is computed, and control is transferred to compute the distance to the intersection with the side of the cone.

The statement

```
  155 IF(DISC.LT.0.0)GOTO 500
```

is used to determine if the value of $\lambda^2-\mu\tau$ is negative. If it is, no intersection with the TEC can occur.

The statements

```
C
C21    SOLVE FOR TWO INTERSECTS WITH QUADRATIC SURFACE
C
       DISC=SQRT(DISC)
       T1=(AMBDA-DISC)/DEN
       T2=(AMBDA+DISC)/DEN
       S1=BETA+ALPHA*T1
       S2=BETA+ALPHA*T2
```

are used to compute $\sqrt{\lambda^2-\tau\mu}$ and, using this value to solve Equations (104) and (105) for θ. These values are used to solve Equation (106).

The statement

```
       IF(WDN.GE.0.0)GOTO 160
```

is used to determine the direction of the ray with respect to the TEC by evaluating $\overline{WB}\cdot\overline{N}$.

The statements

```
       T=S1
       S1=S2
       S2=T
```

are used to interchange the computed values of RIN and ROUT if it was determined by the last instruction that the direction of the ray is negative with respect to the normal of the TEC.

The statements

```
C
C22   DETERMINE IF SIDE INTERSECTION BETWEEN PLANES
C
  160 F=S1*WDN-VXBDN
      IF(F.LT.0.0)GOTO 170
      IF(F.LE.HDN)GOTO 180
  170 S1=-PINF
```

are used to compute the vertical distance etween the plane of the base ellipse and the plane of the first intersect ellipse computed by Equation (106). This distance is evaluated to determine if the intersect occurs between the plane of the base ellipse and the plane of the top ellipse. If not, the intersection distance is set to an extremely large negative value.

The statements

```
  180 F=S2*WDN-VXBDN
      IF(F.LT.0.0)GOTO 190
      IF(F.LE.HDN)GOTO 200
  190 S2=PINF
```

are used to compute the vertical distance between the plane of the base ellipse and the plane of the second intersect ellipse computed by Equation (106). This distance is evaluated to determine if the intersect occurs between the plane of the base ellipse and the plane of the top ellipse. If not, the intersection distance is set to an extremely large positive value.

The statement

```
C
  200 IF(WDN) 220,210,230
```

begins the calculations for determining the intersections of a ray with the plane of the base ellipse and the plane of the top ellipse. This statement is used to determine the direction of the ray with respect to the planes of the TEC.

The statements

```
C
C23   RAY PARALLEL TO PLANES
C
  210 S1=-PINF
```

```
      SO=PINF
      GOTO 300
```

are executed if the ray is parallel to the two planes of the TEC. Therefore, the entry intersect with a plane surface is set to an extremely large negative value, and the exit intersect with a plane surface is set to an extremely large positive value.

The statements

```
C
C24   COMPUTE INTERSECTIONS WITH PLANE SURFACES
C
  220 SI=BETA*ALPHA
      SO=BETA
      LI=2
      LO=1
      GOTO 240
```

are executed for a ray moving in a general direction from the top to the bottom plane. These statements, therefore, compute the distance along the ray to the top and bottom planes, respectively. The surface number assignments are also made for a ray entering the plane of the top ellipse and exiting from the plane of the base ellipse.

The statements

```
C
  230 SI=BETA
      SO=BETA*ALPHA
      LI=1
      LO=2
```

are executed for a ray moving in a general direction from the bottom to the top plane. These statements, therefore, compute the distance along the ray to the bottom and top planes, respectively. The surface number assignments are made for a ray entering the plane of the base ellipse and exiting from the plane of the top ellipse.

The statement

```
  240 IF(SO.LT.0.0) GOTO 500
```

is used to determine if the ray misses the top or base planes by evaluating the distance to the plane that the ray exits. If the distance is negative, the ray is moving away from the TEC.

The statements

```
C
C25   DETERMINE WHICH SURFACE IS HIT
C
  300 IF(SI.GE.S1)GOTO 310
      IF(ABS(SI-S1).LE.0.0001)GOTO 310
      RIN=S1
      LRI=3
      GOTO 350
```

begin the computations to determine if the ray intersections with the TEC occur at the quadratic surface or at a planar surface. A test is first performed to determine if the entry intersection is on a plane surface or on the quadratic surface. If entry is on a plane surface, a branch is made to record RIN and the entry surface number. If the entry occurs on a side, a test is made to determine the difference between the potential entrance on a plane and the potential entrance on the quadratic surface. If the intersections with the plane surface and the quadratic surface occur simultaneously, a branch is made to record RIN and the entry surface number for the plane surface. If not, RIN and the entry surface number are recorded for the quadratic surface. A branch is then executed to compute ROUT.

The statements

```
  310 RIN=SI
      LRI=LI
```

are executed if the previous tests determined that the ray first enters a plane surface. RIN and the entry surface number are assigned from previously computed values.

The statements

```
350 IF(SO.LE.S2) GOTO 360
    IF(ABS(SO-S2).LE.0.0001)GOTO 360
    ROUT=S2
    LRO=3
    GOTO 400
```

are used to determine if the exit intersection occurs on a plane surface or on a side of the TEC. If the exit is on a plane surface, a branch is made to record ROUT and the exit surface number. If the exit intersection occurs on a side, a test is made to determine if the intersection with the plane surface and side surface occur simultaneously. If it does, a branch is made to record ROUT and the exit surface number for the plane surface. It it does not, ROUT and the exit surface number are recorded for the quadratic surface.

The statements

```
360 ROUT=SO
    LRO=LO
```

are executed if the previous tests determined that the ray first exited a plane surface. ROUT and the exit surface number are assigned from previously computed values.

The statements

```
C
 400 IF(RIN.GE.ROUT)GOTO 500
     IF(ABS(RIN-ROUT).LE.0.0001)GOTO 500
     IF(ROUT.LE.0.0)GOTO 500
```

are used to test the computed values of RIN and ROUT. If RIN is greater than ROUT, or the difference between RIN and ROUT is extremely small, or ROUT is negative, the ray misses the TEC, and a branch is therefore performed to record the miss condition.

The statements

```
C
      S=ROUT
      I=1
```

are used to made an assumption that ROUT is from a plane and to assign one to the variable I for later use in a computed GOTO statement.

The statement

```
      GOTO(420,430,410),LRO
```

is used to branch the program to determine if the exit of the ray from a plane lies within the boundary of the ellipse. The destination of the branch depends upon the surface number from which the ray exits. If ROUT is from the side, a branch is made to verify RIN.

The statements

```
  410 S=RIN
      I=2
```

are used to make an assumption that RIN is into a plane and to assign two to the variable I for later use in a computed GOTO statement.

The statement

```
      GOTO(420,430,480),LRI
```

is used to branch the program to determine if the entry of the ray into a plane lies within the boundary of the ellipse. The destination of the branch depends upon the surface number from which the ray exits. If RIN is into the side, a branch is made around the calculations for the planar intersection.

The statements

```
C
C26   DETERMINE IF INTERSECTION WITH BASE PLANE LIES WITHIN
C     CROSS SECTION OF BASE ELLIPSE
C
  420 F1=S*WDA-VXBDA
      F2=S*WDB-VXBDB
      F=F1*F1/(R1*R1)+F2*F2/(R2*R2)
      IF(F.GT.1.0001)GOTO 500
```

are used to compute the base plane intersect distance (F1) from the center along the semi-major axis and the distance (F2) along the semi-minor axis. These distances are used to solve the equation of the ellipse of the base plane $(F1)^2/(R1)^2 + (F2)^2/(R2)^2$. A test is made to determine if the intersect lies outside the ellipse of the base by evaluating the results of the above equation for greater than one since the general equation of an ellipse is $x^2/a^2 + y^2/b^2 = 1$.

The statement

```
      GOTO(410,480),I
```

is a computed GOTO statement that performs a branch based on whether or not the RIN intersect has been verified.

The statement

```
C
  430 IF(R3.EQ.0.0.OR.R4.EQ.0.0)GOTO 480
```

is used to determine if the plane of the top ellipse has any area.

The statements

```
C
C27   DETERMINE IF INTERSECTION WITH TOP PLANE LIES WITHIN
C     CROSS SECTION OF TOP ELLIPSE
C
      F1=S*WDA-VXBDA-HDA
      F2=S*WDB-VXBDB-HDB
      F=F1*F1/(R3*R3)+F2*F2/(R4*R4)
      IF(F.GT.1.0001)GOTO 500
```

are used to compute, for the top plane, the distance (F1) of the intersect along the semi-major axis and the distance (F2) of the intersect along the semi-minor axis. These distances are used in the equation for an ellipse to determine if the intersect lies outside or inside the ellipse of the top plane.

The statement

```
      GOTO(410,480),I
```

performs a branch depending on whether or not the RIN intersect has been verified.

The statements

```
C
C28   RAY ORIGINATES WITHIN TEC
C
  480 IF(RIN.GT.0.0001)RETURN
      RIN=-.0001
      LRI=0
      RETURN
```

test RIN for a positive distance. If RIN is a positive distance, control is returned to the calling program. If RIN is negative, the ray originates within the TEC. RIN is therefore set to a small negative value and the surface entrance number is set to zero. Control is then returned to the calling program.

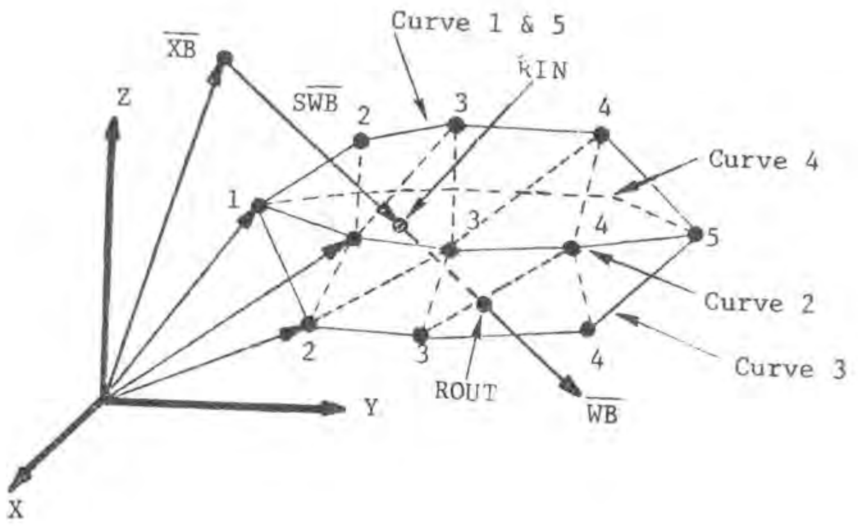
The statements

```
C
C29   RAY MISSES TEC
C
  500 RIN=PINF
      ROUT=-PINF
      LRI=0
      LRO=0
      RETURN
      END
C
C
```

are used if the ray misses the TEC. RIN is set to an extremely large positive value and ROUT is set to an extremely large negative value and returned to the calling program.

Subroutine ARS

This subroutine is used to compute the intersection of a ray with an arbitrary surface, where the arbitrary surface is defined by a specified number of curves and a specified number of points on each curve. The figure described must be closed and solid as shown below:



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 70. Arbitrary Surface

The statements

```
      SUBROUTINE ARS
C
C          SUBROUTINE COMPUTES INTERSECTIONS OF RAY WITH ARBITRARY
C          SURFACE - ARS
C
      DIMENSION W(3),UW(3),VW(3),WXB(3),WN(3),
     1     HIT(20),ORMAL(3,20),ISURF(20)
      DIMENSION MASTER(10000)
      COMMON ASTER(10000)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGO,
     1     LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
      COMMON/DAVIS/IGRID,LOOP,INORM
      COMMON/WHICH/NBO
```

are used to dimension arrays and pass information into and out of this subroutine.

The statement

```
      EQUIVALENCE (MASTER,ASTER)
```

is used to set the MASTER array equivalent to the ASTER array.

The statements

```
C
  901 FORMAT(1H0,12HERROR IN ARS,I5,4X,22HNUMBER OF HITS .GT. 20)
  902 FORMAT(1H0,12HERROR IN ARS,I5,4X,21HNUMBER OF HITS IS ODD,
     1           2X,6H(NHIT=,I5,1H) )
  903 FORMAT(1H0,12HERROR IN ARS,I5,4X,27HWRONG SEQUENCE IN HIT TABLE,
     1           2X,6H(NHIT=,I5,1H) )
  910 FORMAT(5X,46HTHIS ERROR USUALLY MEANS THE ARS IS NOT CLOSED /
     1        9X,3HHIT,5X,7HSURFACE / (F12.4,I12) )
```

are used to format output messages which will occur in the subroutine if certain errors are detected. The following comments explain the data storage of the ARS.

```
C
C          ARS DATA STORAGE IN ASTER ARRAY -
C
C     LOCARS
C       +0       NP   - NUMBER OF POINTS
C       +1       NHIT - NUMBER OF HITS
C       +2       )
C        .       )      - RESERVE 80 WORDS FOR HITS (4 PER HIT)
C        .       )      - ALLOWS FOR 10 PAIRS OF RIN/ROUT
C      +81       )
```

```
C         LOCHTS = LOCARS + 2
C            +0       S      - DISTANCE FROM START POINT XB TO TRIANGLE HIT
C            +1       NX     - DIRECTION COSINES OF NORMAL TO TRIANGLE HIT
C            +2       NY     -    (NX,NY,NZ)
C            +3       NZ     -
C       +82       X      - 4 WORDS PER POINT (X,Y,Z)
C         .       Y      - NP IS TOTAL NUMBER OF POINTS
C         .       Z      -
C         .       FLAG - SET = -1 TO SIGNAL LINE OR POINT TRIANGLE
C          LOC   = LOCARS + 82
C           +0      X
C           +1      Y
C           +2      Z
C           +3      FLAG
C
C          LOC   = LOC   + 4 FOR NEXT TRIANGLE
C
```

The statements

```
C
C          DETERMINE IF RE ENTRY
C
       LOCARS=MASTER(LOCDA)
       LOCHTS=LOCARS+2
       IF(LOOP.NE.KLOOP)GOTO 100
```

are used to set LOCARS to the first word of the data for the ARS. LOCHTS is set to the first storage word of 80 words reserved for hit data. A test is then made to determine if the intersects for the current ray have already been computed from a previous entry to this subroutine. If not, the subroutine branches to clear the hit table and compute the hits for the current ray.

The statements

```
C
C          REENTRY
C
      NHIT=MASTER(LOCARS+1)
      IF(NHIT.LE.0)RETURN
      LOC=LOCHTS
C
C          MOVE INTERSECT DATA TO HIT ARRAY
C
      DO 10 I=1,NHIT
      HIT(I)=ASTER(LOC)
      LOC=LOC+4
   10 CONTINUE
      GOTO 600
```

are executed if the intersects for the current ray have already been computed from a previous entry to this subroutine. These statements are used to locate the number of hits for the current ray. If there were no hits on the ARS for the current ray, the subroutine returns control to the calling program. If there were hits, LOC is set to the starting location of the hit data. The distance from the current origin of the ray, XB, to each intersect of the ARS is stored in the HIT array by the DO loop. The program then branches to determine the correct RIN/ROUT pair for the current ray origin, XB.

The statements

```
C
C         NOT A REENTRY - ZERO INTERSECT DATA SECTION OF ASTER ARRAY
C
  100 NHIT=0
      N=1
      IF(NASC.EQ.-2)GOTO 400
      L1=LOCHTS
      L2=LOCHTS+79
      DO 110 L=L1,L2
      ASTER(L)=0.
  110 CONTINUE
```

are executed if the intersects for the current ray have not already been computed from a previous entry to this subroutine. The number of hits counter is initialized to zero, and a test is made to determine if the normal to the intersect is to be computed. If not, the hit table for storing hits is initialized to zero.

The statements

```
C
C         COMPUTE FOR HIT ON TRIANGLE IT.  STORE FLAG AT ASTER(LOC+3)
C             -1 TRIANGLE IS A POINT OR LINE (DE-GENERATE)
C              0 NON-DEGENERATE, COMPUTE INTERSECT DATA
C
  400 LOC=LOCARS+82
      NT=MASTER(LOCARS)-2
```

are used to set LOC to the beginning location of the ARS point data and to set NT to the total number of possible triangles formed by the ARS points.

The statements

```
      DO 499 IT=1,NT
      IF(ASTER(LOC+3).LT.0.0)GOTO 490
```

are used to begin a DO loop to process each possible triangle to determine if it is intersected by the ray. The test is made to determine if the present set of three points form a degenerate triangle. If they do, the program branches to increment to the next set of three points.

The statements

```
       W(1)=ASTER(LOC)
       W(2)=ASTER(LOC+1)
       W(3)=ASTER(LOC+2)
       UW(1)=ASTER(LOC+4)-W(1)
       UW(2)=ASTER(LOC+5)-W(2)
       UW(3)=ASTER(LOC+6)-W(3)
       VW(1)=ASTER(LOC+8)-W(1)
       VW(2)=ASTER(LOC+9)-W(2)
       VW(3)=ASTER(LOC+10)-W(3)
C          WN = (U-W) X (V-W)
       WN(1)=UW(2)*VW(3)-UW(3)*VW(2)
       WN(2)=UW(3)*VW(1)-UW(1)*VW(3)
       WN(3)=UW(1)*VW(2)-UW(2)*VW(1)
C          D = WB . WN
       D=WB(1)*WN(1)+WB(2)*WN(2)+WB(3)*WN(3)
```

are executed if the present three points form a non-degenerate triangle. These statements compute the vector from the first point to the second point and the vector from the first point to the third point. The cross product of these vectors is then computed which results in a normal vector to the triangle. With the x, y, and z components of $\overline{U} - \overline{W}$, $\overline{V} - \overline{W}$, and $\overline{WB}$ the denominator of Equation (121) is solved.

$$\boxed{\begin{aligned} \alpha(\overline{U}_x - \overline{W}_x) + \beta(\overline{V}_x - \overline{W}_x) - S \cdot \overline{WB}_x &= \overline{XB}_x - \overline{W}_x \\ \alpha(\overline{U}_y - \overline{W}_y) + \beta(\overline{V}_y - \overline{W}_y) - S \cdot \overline{WB}_y &= \overline{XB}_y - \overline{W}_y \\ \alpha(\overline{U}_z - \overline{W}_z) + \beta(\overline{V}_z - \overline{W}_z) - S \cdot \overline{WB}_z &= \overline{XB}_z - \overline{W}_z \end{aligned}} \tag{121}$$

For a point $\overline{XP}$ along the ray to be within the computed triangle

$$\overline{XP} = \overline{XB} + \overline{WB} \cdot S = \alpha\overline{U} + \beta\overline{V} + \gamma\overline{W} \tag{119}$$

where

$$\alpha + \beta + \gamma = 1$$

and

$$0 \leq \alpha \leq 1$$

$$0 \leq \beta \leq 1$$

$$0 \leq \gamma \leq 1$$

Substituting $\gamma = 1 - \alpha - \beta$ into Equation (119), and then forming a matrix to solve for α, β, and S results in the matrix Equation (121).

The statements

```
C
C          DETERMINE IF RAY PARALLEL TO PLANE OF TRIANGLE
C
      IF(ABS(D).LE.0.0001)GOTO 490
      WXB(1)=W(1)-XB(1)
      WXB(2)=W(2)-XB(2)
      WXB(3)=W(3)-XB(3)
```

are used to determine if the value of the denominator of matrix Equation (121) is very nearly zero, which would indicate that the ray is parallel to the plane of the triangle, resulting in a miss condition. If no miss condition, the x, y, and z components of $\overline{W} - \overline{XB}$ are computed.

The statements

```
C          DALPHA = (W-XB) . ( WB X (V-W) )
      DALPHA= WXB(1)*(WB(2)*VW(3)-WB(3)*VW(2))
     1         +WXB(2)*(WB(3)*VW(1)-WB(1)*VW(3))
     2         +WXB(3)*(WB(1)*VW(2)-WB(2)*VW(1))
      ALPHA=DALPHA/D
      IF(ALPHA*(1.-ALPHA).LT.0.0)GOTO 490
```

are used to solve matrix Equation (121) for α, and then to test α to determine if $0 \leq \alpha \leq 1$, which is a condition for the ray to intersect the triangle.

The statements

```
C          DBETA = (W-XB) . ( (U-W) X WB )
      DBETA= WXB(1)*(UW(2)*WB(3)-UW(3)*WB(2))
     1        +WXB(2)*(UW(3)*WB(1)-UW(1)*WB(3))
     2        +WXB(3)*(UW(1)*WB(2)-UW(2)*WB(1))
      BETA=DBETA/D
      IF(BETA*(1.-BETA).LT.0.0)GOTO 490
```

are used to solve matrix Equation (121) for β, and to test β to determine if $0 \leq \beta \leq 1$, which is a condition for the ray to intersect the triangle.

The statements

```
C
      GAMMA=1.-ALPHA-BETA
      IF(GAMMA*(1.-GAMMA).LT.0.0)GOTO 490
```

are used to solve for γ from the relationship $\gamma = 1 - \alpha - \beta$ and then to determine if $0 \leq \gamma \leq 1$, which is a condition for the ray to intersect the triangle.

The statements

```
C
C          COMPUTE DISTANCE TO INTERSECT WITH TRIANGLE
C
C          DS = (W-XB) . WN
C
      DS=WXB(1)*WN(1)+WXB(2)*WN(2)+WXB(3)*WN(3)
      S=DS/D
      CALL UNIT(WN)
```

are used to solve matrix Equation (121) for S, the distance from the origin of the ray to the intersect of the triangle. The direction cosines of the normal to the triangle are then computed by calling Subroutine UNIT.

The statements

```
C
C          DIRECT NORMAL INTO ARS
C
      IF(IT-(IT/2)*2.EQ.0)GOTO 410
      WN(1)=-WN(1)
      WN(2)=-WN(2)
      WN(3)=-WN(3)
      D=-D
  410 JSURF=IT
      IF(D.LT.0.0)JSURF=-JSURF
```

are used to test for an even numbered ARS surface triangle. If even numbered, the normal will be into the ARS as desired. If odd numbered, the normal will be away from the ARS. Therefore, the normal for an odd numbered triangle is reversed along with the denominator value of Equation (121). The intersected surface number is then equated to the ARS triangle number. If the denominator value is negative, the surface number is reversed to indicate an exit intersect. (Actually, the surface number and normal may be opposite to that described above, depending upon how the ARS was defined. This problem is resolved by the statements following statement 540 such that all normals are directed into the ARS.)

The statements

```
C
C          COMPARE NEW INTERSECT DISTANCE WITH DISTANCES ALREADY IN
C          HIT TABLE
C
C          STORE HITS (LARGEST TO SMALLEST)
C
      IF(NHIT.EQ.0)GOTO 430
      DO 420 I=1,NHIT
      IF(ABS(S-HIT(I)).LE.0.0001)GOTO 470
      IF(S.GT.HIT(I))GOTO 450
  420 CONTINUE
```

are used to determine if there are any other hits in the hits table. If the present intersect is not the first hit, the DO loop is executed to either find an intersect with an equivalent distance, or to find the location for the next intersect in the hit table. Intersects are stored largest to smallest in the HIT array.

The statements

```
C
 430 NHIT=NHIT+1
     I=NHIT
     IF(NHIT.LE.20)GOTO 440
 435 WRITE(6,901)NBO
     WRITE(6,910)(HIT(I),ISURF(I),I=1,NHIT)
     NHIT=0
     GOTO 700
```

are used to increment the hit counter by one, if the current hit is the first, or if there is a larger distance already in the hit table. The number of hits counter is equated to variable I, and a test is made to determine if there have been more hits than the capacity of the hits table. If there are more than 20 hits in the hits table, error messages are printed out and the program branches to record a miss condition.

The statements

```
C
C         IF NEW INTERSECT, STORE HIT
C
  440 HIT(I)=S
      ORMAL(1,I)=WN(1)
      ORMAL(2,I)=WN(2)
      ORMAL(3,I)=WN(3)
      ISURF(I)=JSURF
      GOTO 490
```

are used to store the distance to the intersect, the direction cosines of the normal to the intersected triangle, and the surface number of the intersected triangles.

The statements

```
C
C         ADD A HIT TO TABLE WHEN S.GT.HIT(I)
C
  450 J=NHIT
      NHIT=NHIT+1
      IF(NHIT.GT.20)GOTO 435
```

are executed when the current intersect is larger than a hit distance presently in the hit table. These statements save the present number of hits in location J, increment the hits counter by one, and test to determine if the new hit exceeds the capacity of the hit table. If it does, the program branches to print out error messages and to record a miss for the new hit.

The statements

```
C
  460 IF(J.LT.I)GOTO 440
      HIT(J+1)=HIT(J)
      ORMAL(1,J+1)=ORMAL(1,J)
      ORMAL(2,J+1)=ORMAL(2,J)
      ORMAL(3,J+1)=ORMAL(3,J)
      ISURF(J+1)=ISURF(J)
      J=J-1
      GOTO 460
```

are used to insert the current hit to the hit table such that all intersect distances are stored from largest to the smallest distance. The direction cosines array and the intersected surface number array are also rearranged to agree with the order of the intersect distances in the hit table.

The statements

```
C
C         TWO ENTRIES IDENTICAL WHEN S .EQ. HIT(I)
C              IF BOTH RIN OR BOTH ROUT IGNORE
C              IF ONE A RIN AND OTHER A ROUT DELETE ENTRY IN TABLE
C
  470 IF(JSURF*ISURF(I).GT.0)GOTO 490
C
C         DELETE ENTRY
C
      NHIT=NHIT-1
  480 IF(I.GT.NHIT)GOTO 490
      HIT(I)=HIT(I+1)
      ORMAL(1,I)=ORMAL(1,I+1)
      ORMAL(2,I)=ORMAL(2,I+1)
      ORMAL(3,I)=ORMAL(3,I+1)
      ISURF(I)=ISURF(I+1)
      I=I+1
      GOTO 480
```

are executed when an entry in the hit table is found to be equal to the present intersect distance. If the hit table intersect is of the same type as the current intersect (both RIN or ROUT), the program leaves the entry as is and branches to consider the next possible triangle. If the current intersect is different than the entry being tested, the number of hits counter is reduced by one. If there are no additional hits in the table, the program branches to consider the next possible triangle. If there are additional hits in the table, the current entry in the table is eliminated and the hits in the hit table, the normals in the normal array, and the surface number array are rearranged because of the deleted entry.

The statements

```
C
C          INCREMENT TO TEST NEXT POSSIBLE TRIANGLE
C
  490 LOC=LOC+4
  499 CONTINUE
```

are used to increment to the next point in the triangle point data section of the ASTER array, and then branch to test the next three points to determine if the current ray intersects this next triangle.

The statements

```
C
C          ALL POSSIBLE TRIANGLES EXAMINED
C          CHECK FOR AN EVEN NUMBER OF HITS
C
      IF(NHIT.EQ.0)GOTO 700
      IF(NHIT-(NHIT/2)*2.EQ.0)GOTO 500
C
C          ERROR - INCORRECT SEQUENCE OF HITS
C
      WRITE(6,902)NBO,NHIT
      WRITE(6,910)(HIT(I),ISURF(I),I=1,NHIT)
      NHIT=0
      GOTO 700
```

are executed when all possible triangles of the ARS have been examined and are used to test for an even number of hits if any hits on the ARS have taken place. If there were an odd number of hits, error messages are printed out giving the body number and the number of hits. The hit array and the surface number array are also printed out in table format.

The statements

```
C
C          CHECK FOR CORRECT SEQUENCE OF EXITS(-) AND ENTRANCES(+)
C
  500 DO 520 I=2,NHIT
      IF(ISURF(I-1)*ISURF(I).GT.0)GOTO 525
  520 CONTINUE
      GOTO 530
C
C          ERROR - INCORRECT SEQUENCE OF HITS
C
  525 WRITE(6,903)NBO,NHIT
      WRITE(6,910)(HIT(I),ISURF(I),I=1,NHIT)
      NHIT=0
      GOTO 700
```

are used to determine if the hits in the hit table alternate from RIN to ROUT to RIN, etc. If not, error messages are printed out giving the body number and the hit number causing the trouble. The entire hit array and the surface number array are printed out in table format. The number of hits counter is set to zero and the program branches to record a miss of the ARS for the given ray.

The statements

```
C
C          LOCATE NEXT ROUT DISTANCE RELATIVE TO CURRENT POSITION OF XP
C
  530 IF(HIT(NHIT-1).GT.0.0)GOTO 540
      NHIT=NHIT-2
      IF(NHIT.LE.0)GOTO 700
      GOTO 530
```

are used to test the ROUT distances in the hit table, from the smallest to the largest, until a positive distance is found with respect to the current origin of the ray. If no positive ROUT distance is found the program branches to record a miss condition from its present point. When the next ROUT distance is found, the program branches to check if normal distances are to be computed.

The statements

```
C
C          CHECK DIRECTION OF NORMAL FOR LARGEST DISTANCE IN HIT TABLE
C          VERIFY THAT NORMAL IS AN EXIT FOR THE ROUT INTERSECT
C
  540 IF(NASC.EQ.-2)GOTO 600
      IF(ISURF(1).LT.0)GOTO 560
      DO 550 I=1,NHIT
      ORMAL(1,I)=-ORMAL(1,I)
      ORMAL(2,I)=-ORMAL(2,I)
      ORMAL(3,I)=-ORMAL(3,I)
      ISURF(I)=-ISURF(I)
  550 CONTINUE
```

are used to first test if this subroutine was called by Subroutine CALC to compute normal distances. If normal distances are to be computed, the program branches to select the correct RIN and ROUT set. If normal distances are not to be computed, the surface number of the first hit in the hit table is tested to verify that it is negative for a ROUT intersect. If not negative, all of the signs for the normals and surface numbers of the hit table are reversed. This insures that all normals are directed into the ARS and the surface numbers of the intersects have the correct sign.

The statements

```
C
C          STORE HIT TABLE IN ASTER ARRAY
C          UNLESS COMPUTING NORMAL DISTANCE
C
  560 LOC=LOCHTS
      DO 570 I=1,NHIT
      ASTER(LOC)=HIT(I)
      ASTER(LOC+1)=ORMAL(1,I)
      ASTER(LOC+2)=ORMAL(2,I)
      ASTER(LOC+3)=ORMAL(3,I)
      LOC=LOC+4
  570 CONTINUE
```

are used to store the hit table and the normal coordinates (direction cosines) for each of the intersects for the current ray in the ARS section of the ASTER array.

The statements

```
C
C          CHOOSE CORRECT RIN AND ROUT SET FOR CURRENT POSITION OF XP
C          THIS SECTION IS ALSO USED BY REENTRY ROUTINE
C
  600 IF(NHIT.EQ.0)GOTO 700
      RIN=HIT(NHIT)
      ROUT=HIT(NHIT-1)
      LRI=1
      LRO=1
      NHIT=NHIT-2
      IF(ABS(DIST-ROUT).LE.0.0001)GOTO 600
      IF(DIST.GE.ROUT)GOTO 600
      IF(ABS(RIN-ROUT).LE.0.0001)GOTO 600
      IF(RIN.GT.0.0001)GOTO 800
      RIN=-PINF
      LRI=0
      GOTO 800
```

are used to test the RIN/ROUT pairs of intersects against the current distance that the ray has travelled, and to select the RIN/ROUT pair such that RIN $\leq$ Distance $<$ ROUT. If RIN is equal to zero, it is set to a nimus infinity to force the program to choose ROUT.

The statements

```
C
C          RAY MISSES ARS FROM CURRENT LOCATION
C
  700 RIN=PINF
      ROUT=-PINF
      LRI=0
      LRO=0
```

are used to set RIN and ROUT, and the intersected surface numbers, to record a miss condition for the current ray.

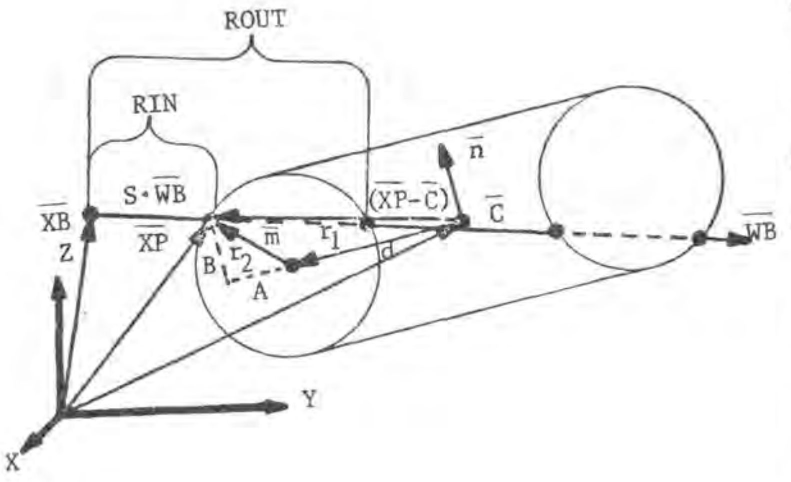
The statements

```
  800 IF(NASC.NE.-2)MASTER(LOCARS+1)=NHIT
      RETURN
      END
```

are used to revise the number of hits in the ASTER array if this subroutine was not called by Subroutine CALL to perform normal intersect calculations. The program then returns control to the calling program.

Subroutine TOR

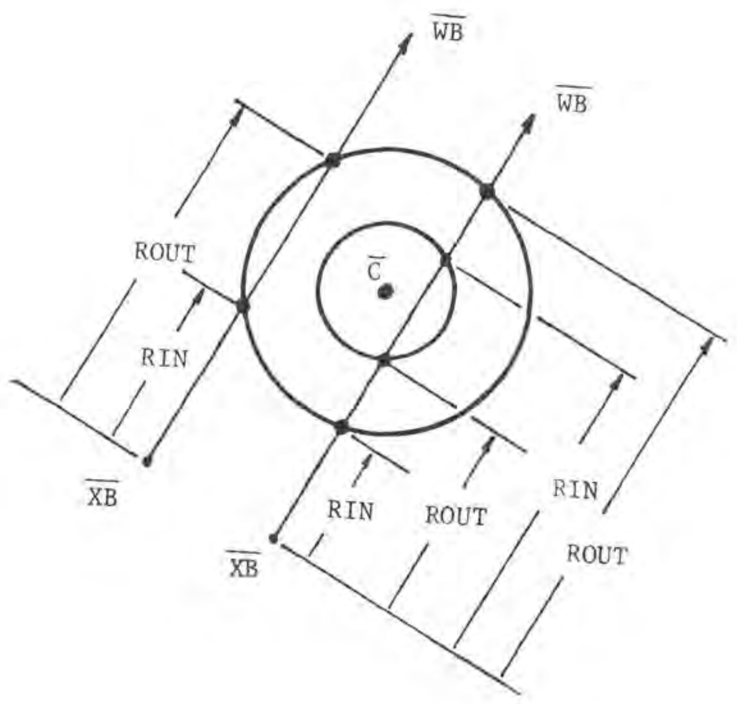
This subroutine computes the ray intersections with a torus. The torus geometry is illustrated in Figure 71.



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{XP}$	point of intersect with torus
$\overline{C}$	center of torus
r_1	radius of torus
r_2	radius of circular cross-section
$\overline{n}$	normal to plane of torus (unit vector)
$\overline{d}$	direction of r_1 (unit vector)
$\overline{m}$	vector from midpoint of cross-section to point of contact

FIG. 71. Torus Geometry

The intersection with the torus may result in either one or two sets of RIN and ROUT as shown in Figure 72.



$\overline{XB}$	starting point of ray
$\overline{WB}$	direction cosines of ray
$\overline{C}$	center of torus
RIN	distance to entry intersect
ROUT	distance to exit intersect

FIG. 72. Intersections with a Torus

If two sets are found, the routine selects the first pair that are greater than the point $\overline{XB}$.

The statements

```
      DIMENSION COEF(4),RT(4),XN(3),XBC(3)
      DIMENSION MASTER(10000)
      COMMON ASTER(10000)
      COMMON/PAREM/XB(3),WB(3),IR
      COMMON/GEOM/LBASE,RIN,ROUT,LRI,LRO,PINF,IERR,DIST
      COMMON/UNCGEM/NRPP,NTRIP,NSCAL,NBODY,NRMAX,LTRIP,LSCAL,LREGD,
     1  LDATA,LRIN,LROT,LIO,LOCDA,I15,I30,LBODY,NASC,KLOOP
      COMMON/DAVIS/IGRID,LOOP,INORM
```

are used to dimension arrays and pass information into and out of the subroutine.

The statement

```
      EQUIVALENCE(ASTER,MASTER)
```

is used to set the ASTER array equivalent to the MASTER array.

The statement

```
C
C1     CHECK FOR PREVIOUS ENTRY
C
      IF(LOOP.NE.KLOOP)GOTO 10
```

is used to check whether the intersections have been computed from a previous entry to the subroutine. If they have not, control is transferred to compute the roots.

The statements

```
C
C2     THIS IS A REENTRY
C
      NR=MASTER(LOCDA+2)
      IF(NR.LE.2)RETURN
      RIN=ASTER(LOCDA+3)
      ROUT=ASTER(LOCDA+4)
      NR=0
      GOTO 400
```

are executed if the intersections for the torus were computed from a previous entry to this subroutine and are used to test whether two sets of RIN and ROUT were obtained. If only one set was computed, the number of real roots variable,NR, will equal 2, and execution will return to the calling routine. If two sets were obtained, the second set is retrieved from the ASTER array and equated to RIN and ROUT. Control is then transferred to Statement 400.

The statements

```
C
C3     RETRIEVE LOCATIONS OF TORUS DATA
C
   10 CALL UN2(LOCDA,IV,IN)
      LOC=LOCDA+1
      CALL UN2(LOC,IR1,IR2)
```

which are executed if ray intersections have not been computed, are used to retrieve the pointers to the vertex, normal, and major and minor axes of the torus in the MASTER array.

The statements

```
C
C4    COMPUTE INTERMEDIATE VARIABLES NEEDED TO
C     FIND COEFFICIENTS OF QUARTIC EQUATION
C
      XBC(1)=XB(1)-ASTER(IV)
      XBC(2)=XB(2)-ASTER(IV+1)
      XBC(3)=XB(3)-ASTER(IV+2)
      XN(1)=ASTER(IN)
      XN(2)=ASTER(IN+1)
      XN(3)=ASTER(IN+2)
      R1=ASTER(IR1)
      R2=ASTER(IR2)
```

are used to compute an intermediate variable for computing the four coefficients to the quartic equation to solve for the <u>int</u>ersections. R1 and R2 are the two radii used in describing the torus. $\overline{XN}$ is the normal to the torus.

The statements

```
      IF(NASC.NE.-2)GOTO 20
      RSAVE=0.
      GOTO 30
```

are used to test the value of NASC. If NASC is set at -1, computations for a new ray are desired. If NASC is set at -2, the routine was called from Subroutine CALC to find the normal distance, and intermediate variable RSAVE is set to zero.

The statements

```
   20 RSAVE=ABS(DOT(XBC,WB))-R1-R2-R2
      XBC(1)=XBC(1)+RSAVE*WB(1)
      XBC(2)=XBC(2)+RSAVE*WB(2)
      XBC(3)=XBC(3)+RSAVE*WB(3)
```

are used to compute intermediate value RSAVE and to temporarily shift vector (XB-V) along the ray to insure correct solution of the quartic equation by Subroutine QRTIC if NASC was set to -1.

The statements

```
30 WDN=DOT(WB,XN)
   XBCDW=DOT(XBC,WB)
   XBCDN=DOT(XBC,XN)
   XBCXBC=DOT(XBC,XBC)
   R1SQ=R1*R1
   R2SQ=R2*R2
   TERM=XBCXBC-R1SQ-R2SQ
```

are used to compute additional intermediate terms required for the coefficients.

The statements

```
C
C*     COMPUTE COEFFICIENTS
C
       COEF(1)=4.*XBCDW
       COEF(2)=4.*R1SQ*WDN*WDN+4.*XBCDW*XBCDW+2.*TERM
       COEF(3)=8.*R1SQ*WDN*XBCDN+4.*XBCDW*TERM
       COEF(4)=4.*R1SQ*XBCDN*XBCDN+TERM*TERM-4.*R1SQ*R2SQ
```

are used to compute the four coefficients required for the quartic solution using Equation (117) where

$$\boxed{\begin{aligned}
A &= 1\\
B &= 4\ \overline{WB}\cdot(\overline{XB}-\overline{C})\\
C &= 4r_1^{\,2}(\overline{WB}\cdot\overline{n})^2 + 4[\overline{WB}\cdot(\overline{XB}-\overline{C})]^2 + 2[(\overline{XB}-\overline{C})^2 - (r_1^{\,2} + r_2^{\,2})]\\
D &= 8r_1^{\,2}(\overline{WB}\cdot\overline{n})[(\overline{XB}-\overline{C})\cdot\overline{n}] + 4[\overline{WB}\cdot(\overline{XB}-\overline{C})]\ [(\overline{XB}-\overline{C})^2 - (r_1^{\,2} + r_2^{\,2})]\\
E &= 4r_1^{\,2}\ [(\overline{XB}-\overline{C})\cdot\overline{n}]^2 + [(\overline{XB}-\overline{C})^2 - (r_1^{\,2} + r_2^{\,2})]^2 - 4\ r_1^{\,2}\ r_2^{\,2}
\end{aligned}} \qquad (117)$$

If the line-of-sight intersects (not the normal distance) are to be computed, the origin of the ray, $\overline{XB}$, is shifted along the direction of the ray, $\overline{WB}$, by an amount, RSAVE, to insure correct solution of the quartic equation by Subroutine QRTIC.

The statement

```
      CALL QRTIC(COEF,RT,NR)
```

is used to call Subroutine QRTIC to solve the quartic equation. The coefficients of the quartic equation are passed in the argument list. Subroutine QRTIC returns with the roots in argument array RT and the number of real roots in argument NR.

The statement

```
C
C6    DETERMINE IF 0, 2, OR 4 ROOTS
C
      IF(NR-2)500,100,200
```

is used to determine if there are no real roots, two real roots, or four real roots.

The statements

```
C
C7    TWO ROOTS
C
  100 IF(ABS(RT(1)-RT(2)).GT.0.0001)GOTO 110
      NR=0
      GOTO 500
```

are executed if the solution of the quartic equation resulted in two real roots. These statements are used to determine if the two intersect distances are very nearly equal and, if they are, to set the number of real roots variable to zero and transfer to record a miss condition.

The statements

```
  110 RT(1)=RT(1)+RSAVE
      RT(2)=RT(2)+RSAVE
      IF(RT(1).LT.RT(2))GOTO 300
      T=RT(1)
      RT(1)=RT(2)
      RT(2)=T
      GOTO 300
```

are executed if the two intersect distances are not very nearly equal. These statements are used to readjust the intersect distances with respect to the original starting position of the ray before it was shifted by RSAVE. The shorter and longer distances are stored in array elements RT(1) and RT(2) respectively, after which the program branches to assign RIN, ROUT, and the surface numbers.

The statements

```
C
C8    FOUR ROOTS
C
  200 DO 210 I=1,4
      RT(I)=RT(I)+RSAVE
  210 CONTINUE
```

are executed if the solution of the quartic equation resulted in four real roots. These statements consist of a DO loop which is used to readjust the four intersect distances with respect to the original starting position of the ray before it was shifted by RSAVE.

The statements

```
C
C9    SORT ROOTS IN ASCENDING ORDER
C
  220 IF(RT(1).LE.RT(2))GOTO 230
      T=RT(1)
      RT(1)=RT(2)
      RT(2)=T
  230 IF(RT(2).LE.RT(3))GOTO 240
      T=RT(2)
      RT(2)=RT(3)
      RT(3)=T
      GOTO 220
  240 IF(RT(3).LE.RT(4))GOTO 250
      T=RT(3)
      RT(3)=RT(4)
      RT(4)=T
      GOTO 230
```

are used to sort the four intersects from the smallest distance to the largest distance in the four-element array, RT.

The statements

```
C
C10   IF RAY TANGENT TO SURFACE ELIMINATE INTERSECTS
C
  250 IF(ABS(RT(2)-RT(3)).GT.0.0001)GOTO 260
      NR=NR-2
      RT(2)=RT(4)
      GOTO 270
  260 IF(ABS(RT(3)-RT(4)).GT.0.0001)GOTO 270
      NR=NR-2
  270 IF(ABS(RT(1)-RT(2)).GT.0.0001)GOTO 280
      NR=NR-2
      RT(1)=RT(3)
      RT(2)=RT(4)
  280 IF(NR.LE.0)GOTO 500
      IF(RT(2).GT.0.0)GOTO 300
      NR=NR-2
      RT(1)=RT(3)
      RT(2)=RT(4)
      GOTO 280
```

are used to compare successive intersect distances — RT(1) with RT(2), RT(2) with RT(3), and RT(3) with RT(4) — to eliminate intersect pairs whose difference is very nearly equal to zero. This indicates that the ray is tangent to the surface. If such a pair is located, the number of roots indicator, NR, is decremented by two, and the remaining roots are stored in the first elements of array RT.

The statement

```
C
  300 IF(NR-2)500,350,310
```

is used to determine the number of remaining intersects after the tangent intersects have been eliminated.

The statements

```
C
C11   FOUR INTERSECTS, DETERMINE WHICH RIN/ROUT SET REQUIRED
C
  310 IF(ABS(DIST-RT(2)).LE.0.0001)GOTO 320
      IF(DIST.LT.RT(2))GOTO 330
```

```
320 RIN=RT(3)
    ROUT=RT(4)
    NR=0
    GOTO 400
330 ASTER(LOCDA+3)=RT(3)
    ASTER(LOCDA+4)=RT(4)
```

are executed if there are four legitimate intersects of the torus. These statements first determine the distance that the point on the ray has travelled relative to the second intersect. If the point on the ray is at the second intersect, the values of RIN and ROUT are determined from the second pair of intersects, and NR, the number of intersects not yet used, is reduced to zero. If the point on the ray is located before the second intersect, the required values of RIN and ROUT will be determined from the first pair of intersects. Therefore, the second pair of intersects is stored in the ASTER array for later use.

The statements

```
350 RIN=RT(1)
    ROUT=RT(2)
```

are executed if there are only two intersects of the torus, or if there are four intersects , but the point on the ray is not yet past the second intersect. These statements therefore assign RIN and ROUT from the first two elements of array RT.

The statements

```
400 LRI=1
    LRO=1
    IF(RIN.GE.ROUT)GOTO 500
    IF(ABS(RIN-ROUT).LE.0.0001)GOTO 500
    IF(ROUT.LE.0.0)GOTO 500
    IF(RIN.GT.0.0001)GOTO 600
    RIN=-.0001
    LRI=0
    GOTO 600
```

are used to set the entering and leaving surface numbers to one and to make a series of tests on the values of RIN and ROUT. If the value of RIN is greater than ROUT, or the absolute difference between RIN and ROUT is very nearly zero, or the value of ROUT is less than or equal to zero, the program branches to record a miss condition. If the value of RIN is very nearly zero or less than zero, RIN is set to a small negative number, and the entering surface number is set to zero to indicate that the ray originates within the torus.

The statements

```
C
C12   RAY MISSES FROM PRESENT ORIGIN
C
  500 RIN=PINF
      ROUT=-PINF
      LRI=0
      LRO=0
```

are executed if the ray misses the torus from the present point on the ray. RIN is set to an extremely large positive number and ROUT to an extremely large negative number. The entering and exiting surface numbers are then set to zero.

The statements

```
  600 MASTER(LOCDA+2)=NR
      RETURN
```

are used to store the number of intersects of the torus with respect to the present distance travelled by the point on the ray. The program then returns control to the calling routine, Subroutine G1 or Subroutine WOWI.

☆U.S. GOVERNMENT PRINTING OFFICE: 1971—769-003/17